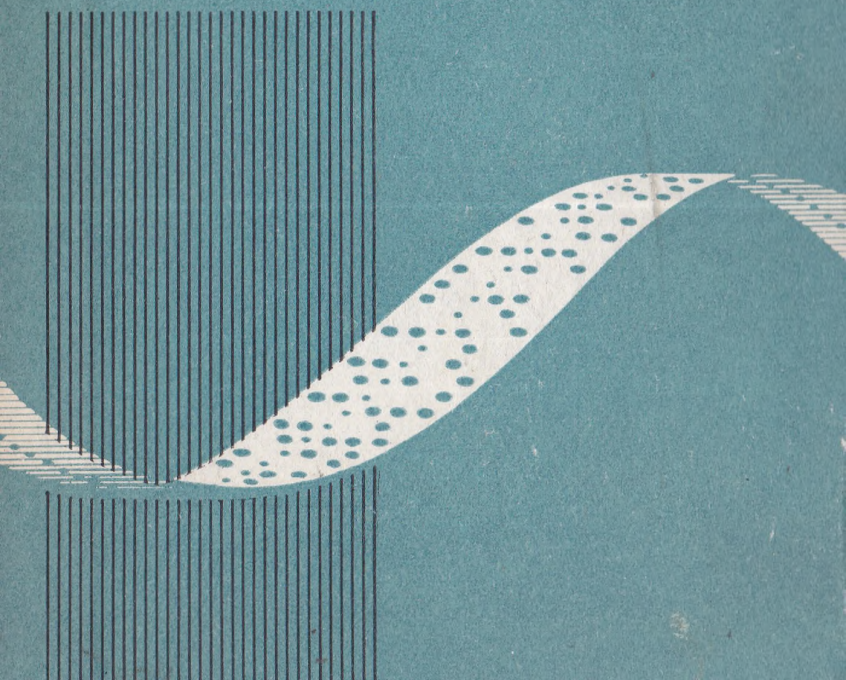


О. С. РАЗУМОВ ★ В. В. ШУРАКОВ

Основные принципы и методы обработки военной информации на ЭЦВМ



О. С. РАЗУМОВ, В. В. ШУРАКОВ

ОСНОВНЫЕ ПРИНЦИПЫ И МЕТОДЫ ОБРАБОТКИ ВОЕННОЙ ИНФОРМАЦИИ НА ЭЦВМ

Ордена Трудового Красного Знамени
ВОЕННОЕ ИЗДАТЕЛЬСТВО
МИНИСТЕРСТВА ОБОРОНЫ СССР
МОСКВА — 1975

6Ф7.3
Р17
УДК 355.233.21

Разумов О. С., Шураков В. В.

Р17 Основные принципы и методы обработки военной информации на ЭЦВМ. М., Воениздат, 1975.
326 с.

В книге изложены основные принципы и методы обработки военной информации на электронно-цифровых вычислительных машинах; приводятся основные сведения о системах управления войсками, о военной информации и ее классификации.

Значительное внимание уделено современным средствам автоматизации обработки информации, структуре и технологическому процессу прохождения задач в типовом вычислительном центре. Материал сопровождается гипотетическими иллюстрациями и примерами.

Предназначается для инженеров-программистов, сотрудников вычислительных центров. Может быть использована курсантами военных училищ, слушателями военных академий и всеми специалистами, работающими в сфере автоматизации управления войсками.

При написании книги использованы материалы, опубликованные в открытой отечественной и зарубежной печати.

Р 30502-173 88-75
068(02)-75

6Ф7.3

© Воениздат, 1975

ВВЕДЕНИЕ

XXIV съезд КПСС в своих программных документах большое внимание уделил коренным вопросам улучшения управления народным хозяйством страны. Съезд определил основные пути решения этой важнейшей социально-политической проблемы:

— совершенствовать структуру управленческого аппарата;

— широко развернуть «работы по созданию и внедрению автоматизированных систем планирования и управления отраслями, территориальными организациями, объединениями, предприятиями, имея в виду создать общегосударственную автоматизированную систему сбора и обработки информации для учета, планирования и управления народным хозяйством на базе государственной сети вычислительных центров и единой автоматизированной сети связи страны. При этом обеспечить с самого начала проведение принципа организационного, методологического и технического единства этой системы» (Материалы XXIV съезда КПСС. Политиздат. М., 1971).

Принципиальные оценки и указания XXIV съезда КПСС стали руководством к действию и для наших военных кадров, которые заняты решением специфических задач управления войсками.

Характер современных боевых действий и условия их проведения предъявляют новые требования к управлению войсками.

Главным явлением процесса развития современных вооруженных сил наиболее развитых стран мира стало принятие на вооружение ракетно-ядерных комплексов. Это вызвало изменения:

- способов и форм вооруженной борьбы;
- организационной структуры войск;
- методов и средств управления войсками.

Скоротечность современного общевойскового боя, необходимость сосредоточения и рассредоточения войск в короткие сроки, непрерывно изменяющаяся обстановка создают сложные условия для управления войсками.

Резкое сокращение времени на принятие решения в этих условиях и увеличение объема работ командиров и штабов по сбору, анализу и оценке данных об обстановке, а также по выполнению оперативно-тактических расчетов требуют дальнейшего совершенствования методов управления войсками.

Одним из наиболее эффективных путей решения этой проблемы является применение современных технических средств для механизации и автоматизации сложных и трудоемких работ, выполняемых командирами и штабами в процессе управления войсками. Поэтому на снабжение войск наиболее развитых государств поступает все большее количество СКМ, СПМ и ЭЦВМ, составляющих техническую основу АСУВ. При создании таких систем широко используются идеи и методы кибернетики как науки об общих законах управления.

Специфичность процессов управления войсками, протекающих в различных условиях, обусловила выделение специальной области кибернетики, получившей название военной кибернетики.

Основным предметом изучения военной кибернетики являются системы и процессы управления войсками, наиболее полно отражающие особенности применения общей теории управления в военном деле.

Главная задача военной кибернетики — совершенствование систем управления войсками с помощью автоматизации наиболее сложных и трудоемких задач, решаемых командирами и штабами в процессе управления. Эту задачу военная кибернетика решает во взаимодействии с тактикой, оперативным искусством, стратегией и другими военно-техническими науками.

Одна из особенностей процессов управления войсками заключается в необходимости учета сознательного, активного противодействия противника.

Этим в основном отличаются системы военного управления от систем управления в других областях чело-

веческой деятельности. Например, при изучении человеком явлений природы она не «скрывает умышленно» от исследователя какую-либо информацию.

В системах управления войсками каждая воюющая сторона всегда стремится не только скрыть от противника сведения, необходимые ему для принятия правильного решения, но и дезинформировать противника. Поэтому решения часто принимаются в условиях неполной и частично ложной информации.

Возникает необходимость разработки особых методов оценки достоверности информации и принятия оптимальных решений.

Специфика вооруженной борьбы отражается и на требованиях, которые предъявляются к системам управления оружием и боевой техникой. Но в методах анализа и синтеза систем управления оружием и боевой техникой и систем управления обычными техническими средствами особого отличия нет.

Процесс управления войсками — сложный творческий процесс, но в нем можно выделить основные функции:

- сбор информации;
- обработка информации;
- принятие решения и доведение его до исполнителей.

Функциональные процессы также состоят из элементарных процессов (действий), которые связаны с решением конкретных задач.

Например, **сбор информации** основан на получении донесений (срочных и вне срочных) от подчиненных и взаимодействующих инстанций, на запросах данных об обстановке (состояние управляемых объектов своих войск, систем противника и условий внешней среды).

Обработка информации сводится к систематизации ее по объектам и определению степени достоверности.

Систематизация осуществляется раздельно по своим войскам и войскам противника, по подчиненным инстанциям и по всей системе управления.

Конечно, полные данные о состоянии всех объектов не будут получены и недостающие данные определяются на основе анализа имеющихся данных с применением методов прогнозирования.

Сбор и обработка информации включают в себя решение большого количества конкретных задач.

Основные из них:

- формирование сообщений о состоянии управляемых объектов, объектов противника и внешней среды;
- передача этих сообщений по линиям связи в управляющий орган;
- фиксация и хранение полученных сообщений;
- формирование запросов на получение данных об обстановке и передача их по линиям связи;
- систематизация информации по объектам и по инстанциям управления;
- подготовка докладов об обстановке командиру и т. д.

В военной кибернетике все эти задачи называются информационными, а информационным процессом — сбор, хранение, обработка и выдача информации.

Принятие решения включает в себя решение ряда задач количественного обоснования принимаемых решений. Например, основные оперативно-тактические расчеты:

- расчет соотношения сил своих войск и войск противника на различных направлениях;
- расчеты необходимого количества средств поражения и средств МТО для выполнения поставленных задач;
- расчет времени на перегруппировку войск и выход их в заданные районы;
- расчет времени развертывания пунктов управления и сетей связи;
- оптимальное целераспределение средств поражения по объектам противника и другие расчеты.

Таким образом, в системах управления войсками решаемые задачи по принадлежности к этапам процесса управления можно разделить на два класса:

информационные (информационно-логические — ИЛЗ);

расчетные (задачи вычислительного характера).

Под автоматизацией процессов управления войсками понимается использование технических средств для оказания помощи органам управления в решении информационных и расчетных задач.

Командир перед принятием решения должен иметь достаточно полные, достоверные и своевременные сведения о данной обстановке. Поэтому важнейшей зада-

чей органа управления (штаба) является получение таких сведений, их обработка и представление результатов в удобном для восприятия виде.

Современный общевойсковой бой характеризуется высокоманевренными действиями и при ручном сборе и обработке информации затраты труда и времени работы офицеров органа управления очень велики. Но данные для принятия решения сильно запаздывают, поэтому принятое решение не обеспечит необходимое качество управления.

На современном уровне развития вычислительной техники, методов формализации задач и программирования автоматизация процесса решения задач вычислительного характера (оперативно-тактические расчеты) принципиально выполняется проще и легче, чем задач информационных.

Вот почему автоматизация организации и обработки больших массивов информации является основой для применения средств автоматизации управления войсками.

ЭЦВМ в автоматизированной системе управления войсками выполняют функции сбора, хранения, обновления и выдачи по запросам различных сведений в период боевых действий.

Комплекс информационных массивов и задач, решаемых в данной автоматизированной системе управления, определяют элементы боевой обстановки, сведения о которых используются при принятии решения.

Для подобных комплексов можно рекомендовать следующие основные задачи:

- по сбору и обработке сведений о своих войсках (состояние, боевые задачи, положение и характер действий);

- по сбору и обработке сведений о противнике;

- по сбору данных о ядерных ударах, радиационной и химической обстановке;

- по подготовке сведений о местности, погоде и т. д. в полосе боевых действий.

Для организации информационных массивов заранее определяются перечень объектов (частей, соединений, органов управления и т. д.) и те характеристики, сведения о которых необходимы данному органу управления.

Эти сведения в памяти ЭЦВМ записываются в виде таблиц, представляющих собой двумерные массивы, т. е. матрицы, по строкам которых ведется запись наименований объектов, а по столбцам — перечень характеристик.

На пересечении строк и столбцов числами, словами или предложениями записываются значения характеристик.

Все объекты объединяются по классам с одинаковыми характеристиками, а таблица для каждого класса записывается в отведенном для нее массиве ВЗУ машины. Каждая строка таблицы дает необходимые сведения об одном объекте и называется его формуляром.

Полнота сведений о боевой обстановке и высокая оперативность их обработки являются важнейшими требованиями, предъявляемыми к автоматизации информационного процесса в любой системе управления войсками.

Выполнение этих требований обеспечивается:

- максимальным упрощением форм входных документов;

- выдачей результатов в форме, удобной для восприятия, или в виде готовых для использования документов;

- небольшим временем поиска данных и их обработки.

Поэтому ключевой проблемой для автоматизации информационного процесса является создание алгоритмических языков, которые удобны для диалога «человек — ЭЦВМ» и для организации хранения, поиска и обработки в ЭЦВМ больших информационных массивов. Конечно, список и характеристики объектов, содержание процесса обработки сведений для каждой информационной задачи будут различными. Поэтому для решения на ЭЦВМ каждой информационной задачи разрабатывается свой алгоритм.

Для автоматизации процесса решения всех информационных задач необходимо разработать комплекс алгоритмов решения следующих служебных задач:

- прием сообщений по каналам связи, их подготовка к обработке на ЭЦВМ и запись в информационное поле МОЗУ;

- перевод текстовой информации с формализованного языка на входной и обратно;

поиск формуляров объектов (по запросам);
сравнение наименования характеристик в таблице и сообщении (обновление);
анализ запросов и выдача справок по ним;
формирование обработанных данных для выдачи;
выдача данных из информационного поля для решения расчетных задач.

Естественно, что связь и необходимая последовательность выполнения алгоритмов решения служебных задач данного комплекса должна выполняться по специальной управляющей программе.

Анализ применения ЭЦВМ в американской армии и отечественного опыта позволяет сделать вывод, что в сфере управления войсками сформировалась информационно-вычислительная служба. Процесс этот закономерен и исторически неотвратим. От того, какие технические средства и программное обеспечение заложены в эту службу, какую квалификацию имеют специалисты ее, зависит информационная вооруженность данной системы управления войсками, вооруженных сил и всей страны.

Таким образом, ЭЦВМ перестали быть «экзотическим предметом», а превратились в действенное средство обработки больших массивов информации. Например, в США производство ЭЦВМ и основанных на них средств автоматизации стало одной из ведущих отраслей промышленности, в которой занято 1,5 млн. человек. В армии США достигнут первый уровень насыщения органов управления ЭЦВМ и для значительной части этих органов ставится вопрос не о приобщении к вычислительной технике, а об углублении применения машин и о замене старого оборудования новым.

По данным американской печати, в США установлено 70 000 ЭЦВМ (журнал «Military Review», 1973, XI). Широкое применение в армии США получили разнообразные системы коллективного пользования (СКП). Вся основная информация по управлению армией США нанесена и поддерживается на машинных носителях. Например, информация системы военного снабжения хранится и эксплуатируется на 240 тыс. полных магнитных лентах.

Применение ЭЦВМ в процессах управления ведет к

изменению всей информационной основы вообще и систем управления войсками в частности.

По мере усложнения ЭЦВМ и расширения их возможностей увеличивается значение **программного** (математического) **обеспечения**. Под этим обеспечением принято понимать комплекс программ регулярного применения — для автоматизации программирования задач, контроля за ходом их решения на машине и организации совместной работы всех устройств ЭЦВМ.

Проблема подготовки специалистов по разработке программного обеспечения, т. е. программистов-системников, является сложной потому, что эта специальность массовая и новая.

Например, по оценке американской печати, в 1975 году США потребуется 400 тыс. программистов-системников. Поэтому необходимо сосредоточить внимание на создании единой информационной базы и соответствующих процедур обработки больших массивов информации, на комплексных задачах в рамках систем управления. При внедрении ЭЦВМ в процессы управления войсками можно сконцентрировать вычислительные и квалифицированные людские мощности для решения наиболее важных проблем. Но этого недостаточно для широкого внедрения машин в сферу управления войсками при использовании их персоналом средней квалификации.

Специалистам необходимо помнить, что XXIV съезд КПСС поставил задачу создания общегосударственной автоматизированной системы управления (ОГАС). Эта система должна объединить все ведомственные системы управления. Такой характер ОГАС, огромные масштабы предстоящих работ по ее созданию выдвигают эту проблему на уровень важнейших, имеющих общенациональный характер.

ГЛАВА I

ОСНОВНЫЕ ХАРАКТЕРИСТИКИ ВОЕННОЙ ИНФОРМАЦИИ, ПРИНЦИПОВ И СРЕДСТВ ЕЕ ОБРАБОТКИ

1.1. ОБЩИЕ ПОНЯТИЯ

Процессы управления характерны для разнообразной деятельности людей. Они проявляются в различных системах управления — от простейших устройств регулирования уличным движением до сложнейших систем управления подвижными объектами, от управления отдельными снарядами до автоматизированного управления боевыми действиями войск.

Эффективное управление возможно, если решения на управление и их реализация основаны на кибернетике, науке об общих закономерностях систем и процессов управления.

Кибернетический закон необходимого разнообразия утверждает, что успешно справиться с разнообразием в управляемом объекте (системе) может только такое управляющее устройство (система), которое само обладает достаточным разнообразием. Поэтому современные системы управления оружием и техникой превратились в сложные человеко-машинные системы.

Системы и процессы управления исследуются и изучаются различными науками. Удалось выявить, что процессы управления в технических системах, живых организмах и общественных системах имеют определенную общность, несмотря на различную материальную природу и методы реализации. Общность проявляется в иден-

тичности структурных схем управляющих устройств (систем) и функций, которые выполняются элементами системы управления. Следовательно, процессы управления в различных системах управления можно идентифицировать одними и теми же математическими зависимостями. Исследования различных процессов управления привели к выводу, что основу всякого управления составляют три части:

- информация о цели и результатах управления;
- анализ информации от объекта управления и принятие решения о необходимых управляющих действиях;
- исполнение принятого решения [28].

Для организации процесса управления в любой системе управления необходимо иметь:

- источники информации о задачах и результатах управления;
- устройства для анализа получаемой информации и выработки решения;
- исполнительные устройства для управления объектами (системами).

В процессах управления часто решающую роль играет информация о результатах управления, поэтому принцип управления с использованием обратной связи является основой при организации процессов управления. Характерным свойством процессов управления является использование в системах управления информации в сигнальной форме. При этом управление является переработкой информации в сигналы, корректирующие деятельность объекта (системы) управления.

При изучении любого процесса управления необходимо рассматривать совместную работу (взаимодействие) управляющего устройства (системы) и объектов управления, образующих сложную динамическую систему. Системой управления называется совокупность всех устройств, которые обеспечивают управление некоторым объектом (системой).

Системы управления, в которых сложный процесс управления осуществляет человек с помощью различных автоматических устройств, называются автоматизированными (АСУ).

Примером АСУ может служить система управления боевыми действиями войск, которая включает комплекс автоматических устройств, вычислительных машин и лич-

ного состава для сбора, передачи и переработки информации в результаты, необходимые командиру для принятия решения.

Под управлением боевыми действиями войск принято понимать постоянное руководство командиров и штабов всех инстанций деятельностью войск. Основные задачи, решаемые в процессе управления войсками:

- поддержание постоянной боевой готовности войск;
- планирование боевых действий и постановка задач войскам;
- своевременная передача команд, сигналов и распоряжений непосредственным исполнителям;
- непрерывный контроль за выполнением принятых решений, состоянием войск и корректировка решений по обстановке;
- всестороннее обеспечение боевых действий войск;
- организация взаимодействия между различными видами и родами войск.

Решение этих задач осуществляется системой управления войсками — совокупностью органов (штабов), средств и методов управления.

В условиях современного научно-технического прогресса перевооружение армий крупнейших стран мира ракетно-ядерным оружием привело к революции в военном деле. Произошли качественные изменения в способах вооруженной борьбы и формах организации армий, в средствах и методах управления войсками и оружием.

В связи с этим совершенствование систем управления войсками стало неизбежным.

В структуре и функциях любой системы управления войсками можно выделить некоторые противоречия:

- коллективная (весь штаб) подготовка решения и единоначалие в процессе принятия его;
- централизация и децентрализация управления;
- твердость и гибкость управления;
- необходимость скрытности и максимальной информированности;
- требование наилучших условий для передачи и для переработки информации системой управления и т. д.

Рассмотрим последнее противоречие. Американские военные специалисты считают, что любое промежуточное звено в организационной структуре войск задержи-

вает прохождение информации. Поэтому путь батальон — дивизия — полевая армия короче, чем путь батальон — бригада — дивизия — армейский корпус — полевая армия. Для наилучших условий передачи информации организационная структура войск должна обеспечивать непосредственное подчинение тактических подразделений (батальонов) штабу командования театром военных действий (рис. 1.1). Тогда информация с переднего края будет попадать прямо к командующему, в штабе которого работают более опытные офицеры, чем в штабе полка. Они помогут командиру батальона выбрать рациональное решение.

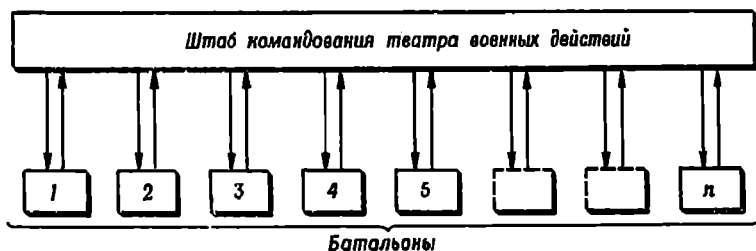


Рис. 1.1. Обобщенная схема наилучших условий для передачи информации

Для наилучших условий переработки информации такая организационная структура неприемлема: ни один штаб с традиционной организацией и техникой обработки информации не сможет в допустимые динамикой современного боя сроки обработать донесения из 200—300 подразделений, оценить обстановку, принять решение и подготовить приказ.

В этом случае для переработки информации войска надо было бы организовать по «двоичному принципу», чтобы каждый командир имел два непосредственно подчиненных подразделения (рис. 1.2). Тогда иерархическая лестница стала бы бесконечной, кроме существующих подразделений появились бы полувзводы, полуроты и т. д. Необходимо разрешить данное противоречие, т. е. найти такое число промежуточных ступеней n_{opt} , при котором суммарное время передачи и переработки информации во всех инстанциях было бы минимальным (рис. 1.3).

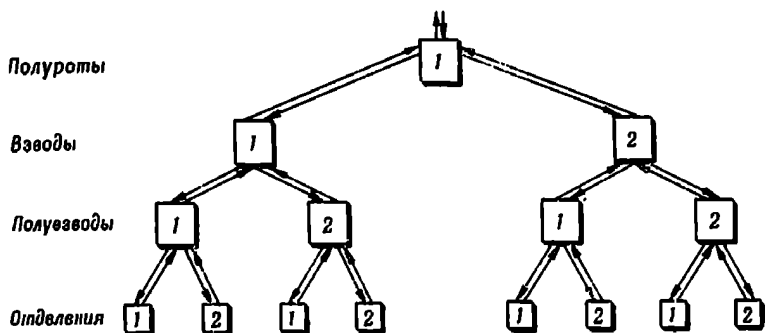


Рис. 1.2. Обобщенная схема наилучших условий для переработки информации

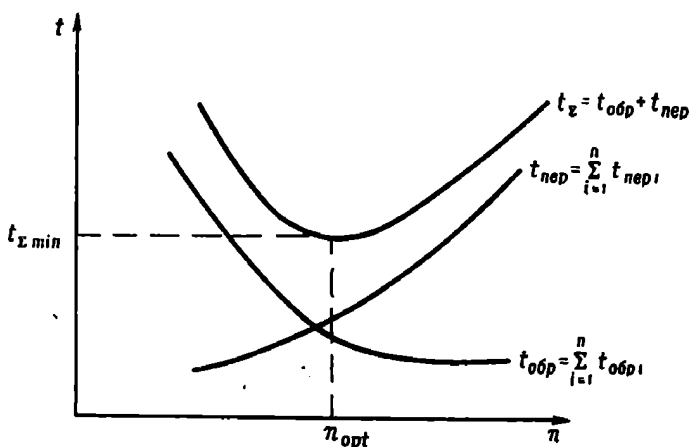


Рис. 1.3. График времени передачи и переработки информации

Например, поиск такой оптимальной структуры управления в армии США, проводимый в пятидесятых годах, вызвал появление «пентомических» дивизий. Заметим, что проблема оптимизации систем управления войсками усложняется с увеличением противоречий и при комплексном подходе к решению. Чем эффективней должна быть система управления, тем труднее разрешить противоречия, тем меньше диапазон допустимых отклонений, т. е. критичность решений резко возрастает при «ужесточении» требований. Общую закономерность можно сформулировать так: повышение эффективности си-

стемы управления прямо пропорционально повышению требований к выбору средств и методов управления. Эти причины привели к появлению специальных разделов по изучению процессов и систем управления в различных науках и созданию кибернетики.

В военной науке возникла теория управления войсками, что нашло отражение в некоторых трудах [11, 28], изданных за последние годы. В теории управления войсками важное место занимает такой гносеологический вопрос, как содержание процесса управления. В научных определениях понятия управления цель отмечается важнейшей характеристикой процесса управления. Содержание процесса управления определяется **целью осуществления (управления)**.

Кибернетика, общая теория систем и процессов управления, рассматривает цель управления обычно как нечто заданное. Для каждой конкретной системы управления цель задается практическими отраслями тех наук, которые исследуют эти системы. Например, цель управления для системы управления полевой артиллерии армии США была поставлена на основе артиллерийской практики военной наукой.

Естественно, что возникла частная задача исследования алгоритмов постановки цели управления человеком для моделирования этого процесса на ЭЦВМ.

Постановка цели управления является прерогативой сознательных существ, создающих в своей голове идеальный образ объективной реальности. Это — чисто человеческая способность.

«Самый плохой архитектор от наилучшей пчелы с самого начала отличается тем, что, прежде чем строить ячейку из воска, он уже построил ее в своей голове. В конце процесса труда получается результат, который уже в начале этого процесса имелся в представлении работника, т. е. идеально»¹. Известно, что в любой системе управления человек — самое сложное и активное звено. Деятельность человека в системах управления включает:

— управление техническими системами (производственные агрегаты, самолеты, системы управления оружием и т. п.);

Маркс К. Капитал. Т. I, Госполитиздат, 1951, с. 185.

— управление социально-экономическими системами (отрасль народного хозяйства, система управления войсками).

Проблема управления социально-экономическими системами пока еще недостаточно изучена. Эти системы включают коллективы людей от небольших групп до отраслей и всего народного хозяйства страны.

Технические средства (ЭЦВМ, связь и т. п.) служат в этих системах промежуточным звеном, они облегчают сбор и обработку информации. Эти системы, как и технические, являются замкнутыми системами с обратной связью, в которых протекают различные динамические процессы. Но в них действует живой человек, который имеет свои цели, что влияет на поведение системы и усложняет задачи управления. Информация (приказы, распоряжения) вводится в эти системы человеком, ее объемы определяются знаниями человека, его личной квалификацией, теми научными методами и средствами управления, которыми он владеет.

В основе всех систем управления лежат объективные закономерности и принципы управления. К основным принципам управления относятся:

- наличие в системе управления и управляющего устройства (органы, система), и управляемого объекта (система);

- использование причинно-следственных и обратных связей между элементами системы;

- взаимодействие системы управления и ее элементов с внешней средой;

- динамический характер системы управления, т. е. способность перехода из одного состояния в другое при сохранении целостности системы.

Каждый процесс управления реализуется посредством передачи и обработки информации (обычно документированной). Каждая система управления состоит из звеньев, осуществляющих сбор, преобразование, анализ, хранение или передачу информации. Поэтому семантически процесс управления является информационным процессом, который состоит из таких операций:

- получение управляющим устройством (системой) по каналу обратной связи информации (состояния) об объекте управления и внешней среде;

- принятие решения, т. е. обработка и преобразование информации состояния;
- постановка задач объекту управления, т. е. передача командой информации по каналу прямой связи.

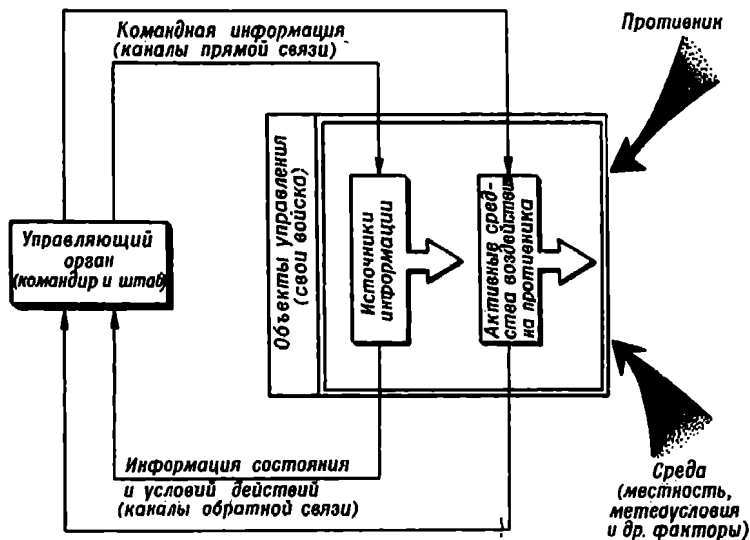


Рис. 1.4. Обобщенная схема замкнутого контура управления в военной системе

Образуется замкнутый контур, в котором управляющее устройство (система) и управляемый объект (система) соединяются каналами прямой и обратной связи (рис. 1.4). Следовательно, процесс управления имеет и информационно-циклический характер, в основе которого лежит циркуляция информации: передача, хранение, обработка (преобразование при выработке решения) и выдача. Разумеется, система управления представляется в виде элементарного замкнутого контура лишь умозрительно в идеализированных условиях. На самом деле управляемый объект (система) находится не в вакууме, а в реальной внешней среде, которая пассивно или активно противодействует достижению цели управления. При этом на каналы прямой и обратной связи воздействуют помехи, а вся система управления может входить в состав более крупной системы (супер-

системы) в качестве подсистемы или управляемого объекта. Кроме того, любая реальная система управления взаимосвязана с подсистемами одной или нескольких суперсистем. Эту схему (рис. 1.4) можно интерпретировать следующим образом:

управляемый объект — воинская часть или подразделение;

управляющее устройство — командир и его штаб;

другие подсистемы — соседи, поддерживающие и обеспечивающие подразделения;

суперсистема — вышестоящая часть или соединение;

внешняя среда — противник (активная, противодействующая среда) и военно-географические условия (пассивная среда).

Естественно, что система управления войсками является сложной динамической, многоступенчатой (иерархической) системой, состоящей из ряда подсистем, она является типичной большой системой.

Известно, что основой успешного проведения боевых действий является правильное решение командира. Управление войсками в бою заключается в своевременном проведении мероприятий, которые обеспечивают подготовку, организацию и выполнение боевых задач подчиненными войсками. Поэтому управление войсками включает следующие этапы:

- сбор сведений об обстановке, ее оценка и выводы в интересах поставленной задачи;
- выполнение необходимых расчетов;
- контроль за действиями исполнителей;
- корректировка решений при изменениях обстановки.

Таким образом, все основные этапы (процесс) управления войсками связаны со сбором, обработкой и передачей информации.

Военной информацией (оперативно-тактической) называются сведения, необходимые для управления войсками [28].

Под информацией вообще понимается отражение в сознании людей тех объективных и субъективных причинно-следственных связей, которые существуют в данной системе управления и в окружающей ее среде. Выражением этих связей являются сведения, идущие от управляемого объекта к управляющему устройству (си-

стеме) и называемые информацией состояния, и команды, идущие от управляющего устройства к управляемому объекту (командная информация). В больших системах управления, в частности в АСУ, потоки информации, поступающие в систему, используются для управления. Такое использование информации, кроме задач обработки и восприятия, требует ее количественной оценки. Сейчас главной задачей теории информации является разработка эффективных методов передачи и обработки (на ЭЦВМ) информации.

Основным вопросом современной теории информации должна быть проблема управления. При этом структура теории такова: перед нами объект (система), которым надо управлять; необходима информация для перевода этого объекта из любого случайного состояния в заданное. Для решения некоторых вопросов, возникающих в процессе управления, недостаточна количественная оценка информации. Для получателя (абонента, пользователя) важно знать ценность переданной информации (сообщения), ее качественную оценку, т. е. необходима разработка методики и аппарата смыслового анализа информации.

Военную информацию (оперативно-тактическую — ОТИ) принято разделять на три вида:

- информация состояния — сведения о численности, вооружении, дислокации противника и своих войск, о районе и условиях боевых действий, характеризующие текущее состояние системы;

- информация о цели управления — информация состояния, обрабатываемая в соответствии с поставленными задачами (целью управления) по правилам (алгоритмам функционирования), изложенным в уставах, наставлениях, с учетом боевого опыта командира, который принял решение;

- **командная информация** (управляющие действия) — команды, сигналы, распоряжения подчиненным войскам в соответствии с решениями, которые принимает командир и штаб на основе обработки информации состояния.

Таким образом, управление войсками при ведении боевых действий определяется переработкой информации состояния в командную информацию по алгоритмам управления (функционирования).

1.2. КЛАССИФИКАЦИЯ

Как уже отмечалось, основой управления войсками является решение командира, обоснованное, принятое на основании изученных сведений о реальном положении войск. Естественно, решения возникают как результат напряженной интеллектуальной работы командира и штаба, связанной с раздумьями, сравнениями, а иногда и моральными колебаниями, что объясняется тем значением, какое имеет решение в достижении цели. Военная информация (ОТИ) объемна и разнообразна по содержанию; она включает:

- данные о войсках противника, своих и войсках соседей;
- данные о радиационной и химической обстановке;
- сведения о местности, районе боевых действий;
- данные о гидрометеорологических условиях и т. д. [28].

Многие данные в условиях современного общевойскового боя быстро устаревают или бывают неполными, противоречивыми, ложными. Вообще военная информация имеет тенденцию к неопределенности. Этим системы управления войсками резко отличаются от систем управления в других сферах деятельности.

Острая конфликтность ситуации, вооруженная борьба двух противников, которая ведется с решительными целями, являются основной особенностью военных динамических систем в отличие от систем других применений [28].

Командир и штаб получают сведения о положении, состоянии и характере действий своих войск и противника, об условиях ведения боевых действий и другие от подчиненных подразделений (частей). Но для принятия решения и процесса управления этих сведений мало. Необходима информация:

- от вышестоящей инстанции (приказы, распоряжения, указания);
- о результатах разведки (всех видов);
- об условиях, характере действий и намерениях соседей;
- о взаимодействии подразделений (частей).

Командир уясняет полученную задачу и оценивает обстановку на основе анализа всей полученной информации, замысла старшего начальника, порядка применения средств поражения на направлениях действий своих войск, определяет место и роль своего подразделения (части) в предстоящем бою.

В результате этого многостороннего, напряженного и неформализуемого процесса принимается решение, которое передается исполнителям и в вышестоящую инстанцию, сообщается соседям и взаимодействующим подразделениям (частям). Сведения, которыми обмениваются командиры и штабы всех степеней в процессе управления войсками, и составляют ядро, основную часть военной информации (ОТИ).

Сведения поступают в документированном и не документированном виде (в приказах, распоряжениях, донесениях, докладах, сводках, различных сообщениях, справках, таблицах, на рабочих картах и т. п.).

Сведения бывают о том, где находятся в данный момент подразделения (части) своих войск, какие боевые действия они ведут, как они обеспечены, их состояние и т. п. Следовательно, управление войсками — это процесс добывания, сбора, технической и творческой обработки, преобразования, передачи, хранения и выдачи (отображения) различной военной информации [11], без которой при самых высоких личных качествах и опыте командир не сможет принять решения.

В результате обработки информации состояния командир получает информацию для исполнителей и старшей инстанции. Большой опыт, деловые качества командира помогают успешнее и быстрее переработать информацию. Информация существует на всех этапах процесса управления и передается материальным носителем.

Одна и та же информация на разных этапах передается разными материальными носителями. Способ передачи и материальный носитель военной информации должны обеспечивать своевременность, скрытность и достоверность передаваемых сообщений.

Классификация военной информации, как и всякая классификация, — условный прием. Она служит для анализа информационных потоков при автоматизации си-

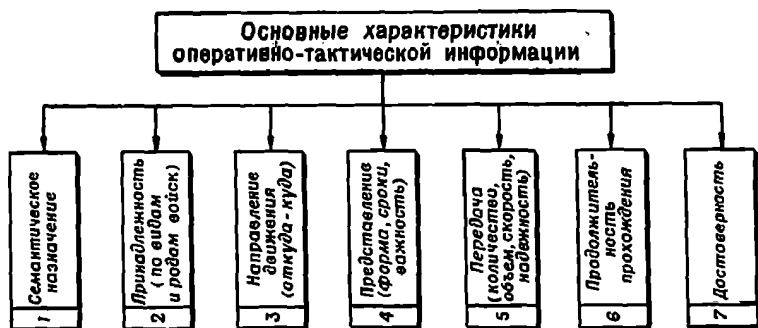


Рис. 1.5. Классификационная схема основных характеристик ОТИ

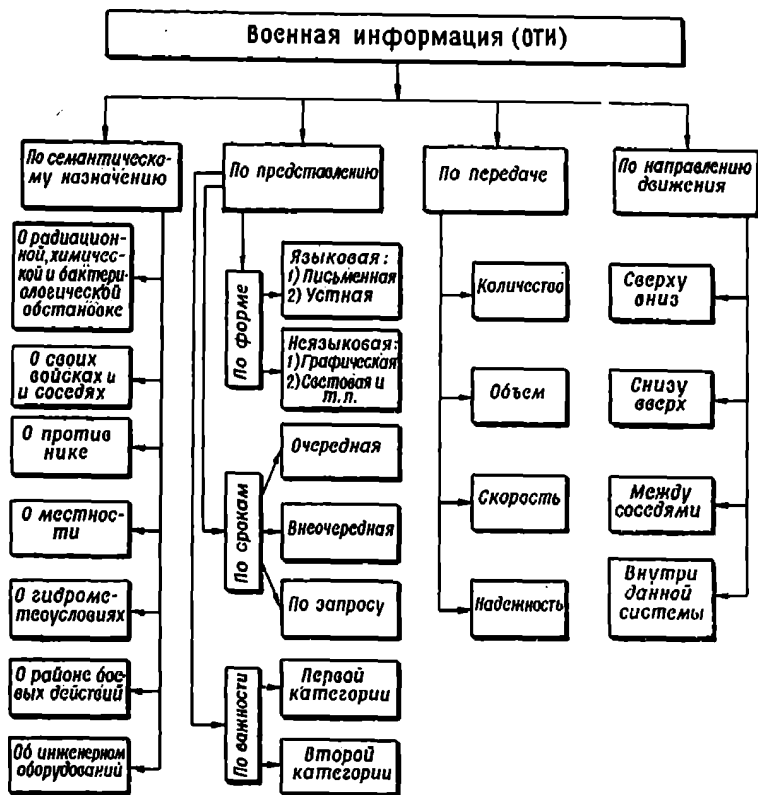


Рис. 1.6. Блок-схема классификации ОТИ

ством управления и определения параметров средств передачи, переработки данных и способов использования этих средств.

Основные характеристики военной информации (ОТИ) американские военные специалисты объединяют в группы (рис. 1.5) классификации. Строгая и единая классификация создает условия для детального описания порядка сбора, обработки и выдачи информации и для сравнительного анализа и определения, где и какими методами целесообразно обрабатывать различные виды ОТИ.

По зарубежным данным классификация основных характеристик военной информации по наиболее важным группам показана на рис. 1.6.

1.3. СРЕДСТВА АВТОМАТИЗАЦИИ

1.3.1. Общие сведения

Как уже отмечалось (1.1), все основные этапы процесса управления войсками: сбор, фиксация и хранение, размножение, передача информации — направлены на обеспечение ее логической обработки.

Главное в управлении войсками — принятие решения. Принятие решения в качестве основного этапа включает логическую обработку информации, осуществляемую людьми в процессе мышления.

Применение вычислительных машин в управлении войсками основывается не только на возможности выполнения ими логической обработки информации, но и на характере управленческой информации.

Важной особенностью военной информации является то, что она содержит очень много различных количественных показателей (чисел). Например, американские военные специалисты опубликовали данные по разработке автоматизированных систем управления войсками, показывающие, что в информации о противовоздушной обороне США или о театре военных действий содержатся миллионы показателей. Поэтому большой удельный вес в системе управления войсками имеет арифметическая обработка этих показателей.

Итак, управление войсками дает вычислительным машинам для обработки огромные информационные потоки (массивы информации).

Современное управление войсками имеет мощную материальную базу — технические средства управления для автоматизации основных этапов процесса управления войсками.

Из большого арсенала технических средств управления войсками выделим один класс, играющий наиболее

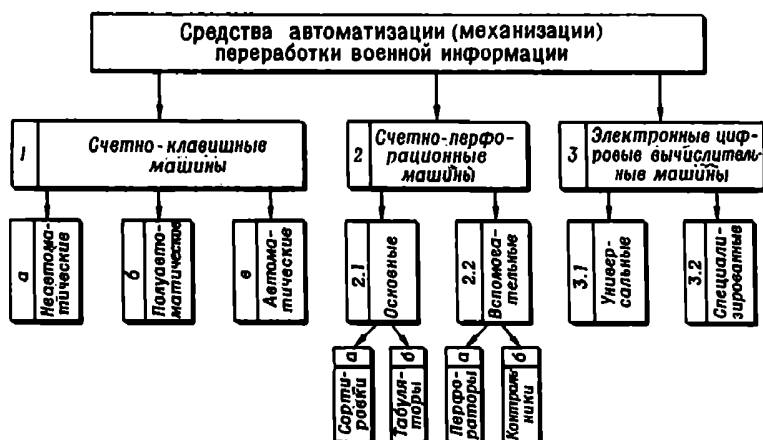


Рис. 1.7. Блок-схема классификации средств автоматизации переработки военной информации

важную роль в процессе управления — средства автоматизации (механизации) переработки военной (ОТИ) информации (рис. 1.7). Эти средства называют или средствами вычислительной техники, или вычислительными машинами.

Выбор вида вычислительных машин на том или ином этапе процесса управления войсками зависит от объема и сложности предстоящей обработки информации и от того, насколько точными должны быть результаты.

1. Счетно-клавишные машины (СКМ)

Эти машины предназначены для механизации арифметических операций. Некоторые из них обеспечивают запись числовой и текстовой информации.

СКМ повышают скорость вычислений по сравнению с вычислениями вручную в 2—4 раза, имеют ручной

ввод исходных данных и ручное управление вычислительными операциями. Они широко применяются для обработки информации материального и технического снабжения войск. По степени механизации и автоматизации процесса обработки информации СКМ разделяются на следующие группы:

- неавтоматические (ручные) — например, десятиклавишная машина ВК-1 и арифмометр «Феликс»;
- полуавтоматические (советские СДМ-107, ВМП-2 и др.);
- автоматические (советские ВММ-2, ВММ-3 ВК-3 и др.).

Средняя производительность советских автоматических СКМ:

- при сложении (вычитании) — до 1000 операций в час;
- при умножении — до 500 операций в час;
- при делении — до 300 операций в час.

За последние годы отечественная промышленность освоила выпуск электронных СКМ («Искра-11, -12, -22», «Рось» и др.) с производительностью до 40—50 тыс. операций в час [30].

2. Счетно-перфорационные машины (СПМ)

СПМ имеют наиболее высокий уровень механизации и автоматизации логической обработки информации. Они представляют собой целую систему взаимосвязанных устройств, в которых информация обрабатывается последовательно.

Свое название СПМ получили от перфокарт, которые применяются в них для ввода и вывода информации. При этом перфокарты можно сортировать и подсчитывать.

Все оборудование СПМ разделяется на две группы: **основное** — для обработки информации (перфокарт), которое состоит из сортировальных машин и табуляторов;

вспомогательное — для подготовки перфокарт к обработке, которое состоит из перфораторов и контрольных.

В настоящее время советская промышленность выпускает электромеханические и электронные устройства

для комплексов СПМ с достаточно хорошими техническими характеристиками: перфоратор — П-80-6/1М; контрольный — К-80-6/1М; электронная сортировка — СЭ-80-3/1М; табулятор — ТА-80-1; итоговый перфоратор — ПИ-80-У; электронная вычислительная приставка — ВП-3 и др. СПМ имеют более высокую производительность, чем СКМ.

К недостаткам СПМ можно отнести:

- большие затраты времени при подготовке перфокарт;
- необходимость участия человека при анализе промежуточных результатов обработки информации;
- необходимость в переносе перфокарт;
- необходимость настройки и перекоммуникации органов управления;
- громоздкую «память».

Возможность классификации данных и подсчет итогов, объединенных общим классификационным признаком, являются неоспоримыми достоинствами СПМ.

3. Электронные вычислительные машины

В этом классе вычислительных машин выделяются два вида — универсальные и специализированные ЭЦВМ. ЭЦВМ существенно отличаются от других технических средств. Применение ЭЦВМ требует принципиально нового подхода. Главное в применении ЭЦВМ связано не столько с эксплуатацией машины, сколько с подготовительными этапами обработки информации: формализацией и алгоритмизацией процессов обработки информации, выбором метода обработки больших массивов информации и программированием решения задач. Эти вопросы изложены в главах 2 и 3.

В истории развития электронных вычислительных машин и их применения принято выделять поколения ЭВМ.

Хронологически 1-е поколение относится к 1952—62 гг.; 2-е поколение — к 1962—1972 гг.; 3-е поколение начинается с 1972 г.

Естественно ЭВМ разных поколений в структурном, техническом, информационном и функциональном отношениях неодинаковы.

Для программистов и специалистов по применению ЭВМ важное значение имеет характеристика математического обеспечения ЭВМ по поколениям (табл. 1).

Математическое обеспечение ЭЦВМ трех поколений

Поко- ление	Типичные задачи	Организация работы	Математическое обеспечение
1-е	Вычислительные	Личный счет	Стандартные програм- мы. Компилирующие и ин- терпретирующие систе- мы. Языки символического кодирования. Отладочные програм- мы
2-е	Обработка эконо- мической информации Обработка сим- вольной информа- ции Обработка эк- спериментов в ре- альном времени	Операторский счет Пакетная обработка	Модульное программи- рование. Проблемно-ориентиро- ванные языки Системы программиро- вания, включающие средства отладки Автооператоры
3-е	Автоматизиро- ванное управление Автоматизиро- ванное проектиро- вание	Многопро- граммная работа Разделение времени Совместная работа чело- века с машиной	Супервизоры Мониторы Универсальные алго- ритмические языки. Программы обслужи- вания вынесенных пуль- тов управления. Информационные ар- хивы

Современной электронной цифровой вычислительной машиной, завершающей семейство машин 2-го поколения «Минск», является «Минск-32». Эта машина наиболее полно отвечает требованиям, предъявляемым к современным серийным ЭЦВМ широкого применения.

В основном машина применяется в вычислительных центрах НИИ, заводов, разнообразных систем управления, в которых она выполняет функции базовой машины системы [25].

Государственный Комитет по науке и технике при Совете Министров СССР принял решение выпускать ЭЦВМ «Минск-32» серийно до конца 1975 г.

В последующих разделах дается краткая характеристика единой системы электронных вычислительных машин (ЕС ЭВМ) 3-го поколения, рассматриваются основные технические характеристики одной из моделей ЕС ЭВМ и основные функции системы программного обеспечения ЕС ЭВМ.

1.3.2. Электронная цифровая вычислительная машина (ЭЦВМ) «Минск-32»

ЭЦВМ «Минск-32» — универсальная, многопрограммная машина средней производительности по обработке алфавитно-цифровой информации. Она предназначена для решения научно-технических, планово-экономических и управленческих задач, а также пригодна, соответственно, для решения задач военно-технических, военно-экономических и обработки документов управления войсками.

Основные особенности ЭЦВМ «Минск-32»:

- большой объем магнитного оперативного запоминающего устройства — МОЗУ (до 65 536 машинных слов);

- возможность многопрограммной работы, что достигается одновременным обменом информацией с несколькими внешними устройствами параллельно с работой вычислителя по одной из рабочих или служебных программ; в МОЗУ одновременно могут храниться восемь служебных и четыре рабочие программы;

- обеспечение защиты одной программы в МОЗУ от другой, т. е. границы защиты задаются с точностью до одного листа (512 ячеек МОЗУ);

- наличие двух каналов связи для подключения к машине до 136 внешних устройств (по универсальным принципам);

- наличие быстрого канала связи для подключения ВЗУ типа магнитных лент, барабанов, дисков;

- возможность одновременной работы вычислителя и внешних устройств быстрого и медленного каналов;

- обеспечение совместимости с ЭЦВМ «Минск-22м» — все программы для «Минск-22м» и на автокоде «АКИ»

для «Минск-2(22)» могут эксплуатироваться без перепрограммирования на «Минск-32»;

— использование в качестве информационных единиц 37-разрядного машинного слова для работы с командами и числами и 7-разрядного символа (код ГОСТ 10859—64) — для работы с полями памяти переменной длины и для обработки массивов информации внешних устройств; распределение разрядов ячейки памяти при записи информационных единиц показано на (рис. 1.8);

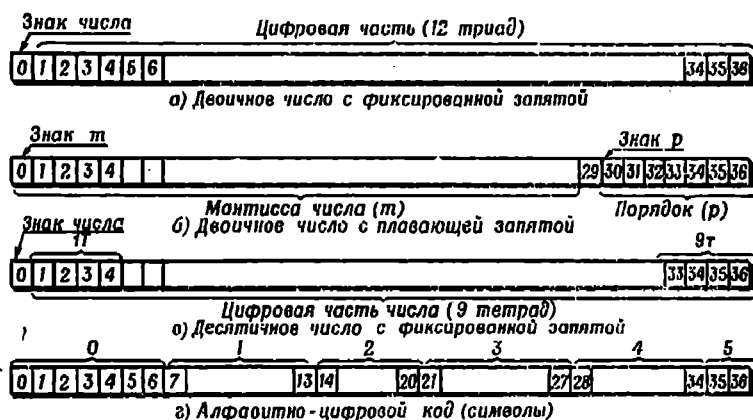


Рис. 1.8. Схема распределения разрядов ячейки памяти при записи информационных единиц

— использование для организации посимвольной обработки информации адресации каждого алфавитно-цифрового символа с указанием номеров ячейки памяти и символа в ячейке (в командах «записи» и «чтения» символов в МОЗУ);

— возможность оперировать двоичными и десятичными числами; при этом двоичные числа могут быть представлены в форме: целых чисел, с фиксированной или с плавающей запятой, а десятичные — в двоично-десятичном коде (система «8421») в форме: целых чисел и с фиксированной запятой;

— возможность выполнения некоторых команд в групповом режиме, что повышает быстродействие машины и сокращает количество команд в программе;

— использование базисного адреса (регистра) для программирования в относительных адресах, чтобы обеспечить рациональное распределение памяти и свободный доступ в любое поле ее; в каждой программе можно использовать до четырех базисных адресов;

— возможность использовать для каждой программы минимум — 16 индексных ячеек памяти (используется 15) — при большем количестве объем памяти всегда должен быть кратен 16;

— наличие программно-аппаратурной службы времени (системы разделения времени);

— возможность совместной работы двух ЭЦВМ.

Структурная схема ЭЦВМ «Минск-32»

Основной комплект оборудования машины состоит из устройств, наименование и назначение которых приведены в табл. 2.

Структурная схема машины (рис. 1.9) показана в общих чертах. Ниже дается краткая характеристика основных устройств вычислителя.

1. **Центральное устройство управления (ЦУ)** предназначено для обеспечения автоматического выполнения команд программы обработки информации. ЦУ синхронизирует работу всех устройств машины, вырабатывает серии тактовых импульсов для выполнения элементарных действий в определенной для каждой операции последовательности, задает режимы работы машины и т. д. В состав устройства входят следующие узлы:

— блок выполнения пуска — останова;

— блок распределения импульсов и переключения тактов;

— дешифратор операция (ДшОп) вместе со своим регистром (РДшОп) и регистром определителя (РОпр) — предназначены для преобразования кода операции, а при необходимости и кода определителя в управляющие сигналы подготовки вычислителя к выполнению данной команды;

— регистр адреса (РА — 16-разрядный) — для приема адреса ячейки при обращении к МОЗУ;

— счетчик адресов команд (СЧАК — 16-разрядный) — для получения адреса очередной выполняемой команды;

Состав и назначение основных устройств комплекта машины

Наименование оборудования	Состав оборудования	Обозначение	Количество в комплекте
Вычислитель (процессор)	Центральное устройство управления	ЦУ	1
	Арифметическое устройство	АУ	1
	Магнитное . оперативное запоминающее устройство	МОЗУ	1
	Устройство обмена	УО	1
	Центральный пульт управления	ЦПУ	1
	Устройство питания вычислителя	УПВ	1
Устройство ввода перфо- ленточное (УВЛ-23) — один комплект	Устройство управления ввода	УУВЛ-23	1
	Фотоэлектрическое входное устройство	FS-1500	1
Устройство вывода перфо- ленточное (УВЛ-23) — два комплекта	Устройство управления вывода	УУВЛ-23	1
	Перфоратор	ПЛ-80/8	1
Устройство ввода	Перфокарточное	УВБК-600М	1
Устройство вывода перфокарточное (УВК-23) — один комплект	Устройство управления вывода	УУВК-23	1
	Перфоратор	ПЭМ-80	1
Устройство вывода печатающее (УПч-23) — один комплект	Устройство управления печатью	УУПч-23	1
	Алфавитно-цифровое печатающее устройство	АЦПУ-128/2м	1
Устройство ввода — вывода пишущей машинки (УМП-23К) — один комплект	Устройство управления машинкой	УУМП-23К	1
	Машинка пишущая	„Консул-254“	1
Запоминающее устройство на магнитной ленте — один комплект	Устройство управления МЛ	УУМЛ-67	1
	Накопитель на МЛ	НМЛ-67	6
Устройство подготовки данных на перфокартах (УПДК)	Стол	УПДК	1
	Перфоратор	П-80/6	1
	Аппарат телеграфный	СТА-М-67	1
	Контрольщик	К-80/6	1
Устройство подготовки данных на перфоленте	Контрольно-считывающее устройство	КСУ-2	2

— два двухразрядных регистра номера базиса (РНБ1 и РНБ2) — для первого (А1) и второго (А2) адресов;

— регистр базиса индекса (РБИ—12-разрядный) и регистр номера индекса (РНИ — 4-разрядный), в которых формируется адрес индексной ячейки;

— два семиразрядных регистра — для хранения адресов начала и конца защиты памяти (РЗ-1 и РЗ-2), которые работают совместно со схемой сравнения с содержимым РА;

— регистр указателей (РУК — 7-разрядный) — для хранения состояния указателей вычислителя;

— регистр индикаторов (РИ — 6-разрядный) — для хранения состояния индикаторов вычислителя;

— регистр номера уровня (РНУ — 4-разрядный).

2. Арифметическое устройство (АУ) — универсального типа, параллельного действия, построенное по трех-регистровой схеме, — обеспечивает:

— выполнение арифметических действий с двоичными и десятичными числами;

— выполнение логических действий с кодами;

— базирование и индексирование (действия с адресными частями команд) и т. д.

В состав АУ входят следующие узлы:

— регистры Р1 и Р2, имеющие по тридцать семь двоичных разрядов;

— сумматор (См) накапливающего типа (38-разрядный) — используется в качестве третьего регистра; может работать в двоичной или десятичной системах счисления; для ускорения операций имеет схему группового переноса; цепь циклического переноса из знакового разряда в младший цифровой для выполнения операции сложения (вычитания) в обратном коде над двоичными числами;

— местное управление (МУ АУ) — согласовывает работу всех узлов устройства при выполнении различных операций.

Узлы АУ связаны между собой и с 37 кодовыми шинами числа (КШЧ) для обмена информацией с МОЗУ. Шесть старших цифровых и один знаковый разряды См (с 0-го по 6-й) и Р2 связаны с внутренними кодовыми шинами символа (КШС) для передачи символов при

операциях над ними и при обмене информацией с внешними устройствами. Выходные цепи См с 5-го по 20-й и с 21-го по 36-й разряды связаны с кодовыми шинами адреса (КША) для передачи в МОЗУ проиндексированных и пробазированных адресов (А1 и А2) команд.

3. Магнитное оперативное запоминающее устройство (МОЗУ). В ЭЦВМ «Минск-32» используется МОЗУ матричного типа на ферритовых сердечниках (ФС). Основу МОЗУ составляет матричный ферритовый куб емкостью в 16 К (К-1024 38-разрядных ячеек).

При этом в каждой ячейке МОЗУ 37 двоичных разрядов являются информационными (кодовыми) и один разряд — контрольный (k). При записи информации в ячейку в этот разряд записывается дополнение до нечетного количества единиц — в информационных разрядах. При считывании информации из ячеек МОЗУ подсчитывается и контролируется общее количество единиц в каждой ячейке (учитывается и контрольный разряд). Четное количество единиц вызывает сбой в работе вычислителя.

Время обращения к любой ячейке МОЗУ при «записи» или «чтении» равно 5 мксек.

Напомним, что обмен информацией с МОЗУ осуществляется 37-разрядными машинными словами или 7-разрядными символами. Обмен символами выполняется по специальным командам или при работе с внешними устройствами.

При записи символа по заданному адресу сначала считывается все содержимое (машинное слово) данной ячейки МОЗУ. Затем в этой ячейке заменяется символ с заданным номером, формируется контрольный разряд и записывается новое слово в МОЗУ по тому же адресу.

При чтении символа сначала считывается машинное слово из данной ячейки, затем из него выбирается заданный символ и машинное слово без изменений записывается (регенерируется) по тому же адресу.

4. Устройство обмена (УО) предназначено для связи между вычислителем и внешними устройствами по двум каналам: быстрому (БК) и медленному (МК).

Через БК подключаются внешние устройства (ВНУ), в которых скорость обмена символами соизмерима с временем обращения к МОЗУ. Например, при работе

с магнитной лентой (МЛ) интервал времени между передачей двух символов равен 16 мксек, а время обращения к МОЗУ равно 5 мксек.

Через МК к вычислителю подключаются ВнУ с меньшей скоростью работы: устройства ввода и вывода на перфоленту и перфокарты, печатающее устройство и пишущая машинка. Например, при вводе информации с перфоленты интервал времени между передачей двух символов составляет около 650 мксек, а при вводе с 80-колодных перфокарт интервал времени между чтением соседних колонок равен 1250 мксек. Внешние устройства БК подключаются к вычислителю группами (0, 1, 2, 3-я) по восемь устройств через четыре коммутатора. Например, лентопротяжные механизмы НМЛ подключаются через коммутатор 0-й группы, который конструктивно монтируется в шкафу УУМЛ-67.

Внешние устройства МК подключаются четырьмя группами (4, 5, 6, 7-я).

Четвертая группа подключает семь устройств основного комплекта: УВВЛ-23, УВЛ-23 (два комплекта), УВвК-23, УВК-23, УПч-23 и УМП-23К. В эту группу может быть включено еще одно устройство МК. ВнУ остальных трех групп подключаются через коммутаторы по 32 устройства в каждой группе. При этом вместо каждого коммутатора может подключаться одно устройство. Всего к вычислителю может быть подключено 32 устройства БК и 104 устройства МК. Вычислитель при одновременном выполнении программ может работать с одним устройством БК (с одной МЛ) и с несколькими ВнУ МК.

В каждом внешнем устройстве медленного канала имеется регистр хранения символов (РС), что создает возможность одновременного обмена символами с несколькими устройствами МК.

При обмене символами с внешними устройствами медленного канала в каждом такте приостановки управляющее слово (УС) из управляющей области памяти (УОП) принимается в регистр управляющего слова (РУС) устройства обмена (УО). Поэтому из этого регистра (РУС) адрес очередного символа (в УС) выдается в регистр адреса (РА) ЦУ для обращения к МОЗУ. По этому адресу происходит «чтение» или «запись» в соответствующую ячейку МОЗУ очередного символа. Ад-

рес же в РУС (из УС) увеличивается на единицу, после чего данное управляющее слово (УС) снова записывается в ту ячейку УОП, из которой оно было принято в РУС. Длительность такта приостановки для ВнУ медленного канала равна 15 мксек, т. е. составляет три такта обращения к МОЗУ.

При обмене символами с внешними устройствами быстрого канала УС заранее пересылается в буферный регистр управляющего слова (БРУС) устройства обмена, где и находится на интервале времени обмена. Поэтому такт приостановки для ВнУ быстрого канала равен 5 мксек, т. е. совпадает с временем обращения к МОЗУ.

5. Центральный пульт управления (ЦПУ) состоит из двух частей:

пульт инженера — оформлен в виде самостоятельного стола;

пульт оператора — оформлен в виде переносного пульта и расположен на столе рядом с пишущей машинкой «Консул-254».

Пульт инженера предназначен для проверки машины, отыскания неисправностей, проведения профилактики и для отладки программ математического (программного) обеспечения; состоит из двух панелей: индикации и управления.

На панель индикации вынесена система сигнальных лампочек для высвечивания состояния отдельных устройств и узлов вычислителя:

- регистров и сумматора АУ (Р1, Р2 и См);
- счетчика адреса команд (СЧАК);
- регистров номера индекса (РНИ);
- кода операции (РДшОп);
- определителя (РОпр);
- номеров базисов (РНБ1 и РНБ2);
- указателей (РУК);
- индикаторов (РИ);
- тактов выполняемой команды.

На панели управления расположены клавиши:

- занесения информации в отдельные регистры вычислителя;
- гашения содержимого регистров вычислителя;
- числового и адресного наборов (с подсветкой);
- блокировок.

Пульт оператора и пишущая машинка «Консул-254» предназначены для обеспечения работы оператора при выполнении ЭЦВМ программ. На пульте оператора имеются различные клавиши управления и индикация остановки и неготовности машины.

Пишущая машинка выдает оператору сообщения и указания в ходе выполнения ЭЦВМ программ и служит для ввода в ЭЦВМ директив оператора.

Ниже приведены характеристики ЭЦВМ «Минск-32».

Основные характеристики ЭЦВМ «Минск-32»

Вычислитель

Система команд	Одно-двухадресная
Форма представления чисел	Двоичная — с плавающей, фиксированной запятой и целая; десятичная — с фиксированной запятой и целая
Разрядность двоичных чисел с плавающей запятой	Мантисса — 28 разрядов плюс разряд знака
Разрядность двоичных чисел с фиксированной запятой	36 разрядов плюс разряд знака
Разрядность десятичных чисел	9 десятичных разрядов плюс разряд знака
Диапазон чисел:	
с фиксированной запятой	От -1 до $+1$
с плавающей запятой	От $\pm 10^{-19}$ до $\pm 10^{+19}$
десятичных	От $\pm 10^{-9}$ до $\pm 10^{+9}$
Емкость оперативной памяти (МОЗУ)	16 384—65 536 машинных слов
Цикл работы МОЗУ	5 мксек
Время выполнения операций:	
сложение двух двоичных 37-разрядных чисел	25—65 тыс. операций в секунду

умножение двух двоичных 37-разрядных чисел	.8—65 тыс. <i>операций в секунду</i>
обмен двух машинных слов	.25 тыс. <i>операций в секунду</i>
передача управления	до 100 тыс. <i>операций в секунду</i>
сложение двух 10-разрядных десятичных чисел	.30—40 тыс. <i>операций в секунду</i>
умножение двух 10-разрядных десятичных чисел	.1,5—40 тыс. <i>операций в секунду</i>

Устройство ввода информации с перфокарт

Носитель информации	.80- или 45-колонные перфокарты
Представление информации	.ГОСТ 10859—64, по- колонное или лю- бое другое с про- граммным переко- дированием
Скорость ввода	.600 карт в минуту
Емкость приемного и подающего карманов	.1000 карт
Тип механизма	.ВУ-600

Устройство ввода информации с перфоленты

Носитель информации	.5 — 8 - дорожечная перфолента
Представление информации	.ГОСТ 10859—64 и любое другое с программным пе- рекодированием
Максимальная скорость ввода	.1500 строк в се- кунду
Тип механизма	.FS-1500

Устройство вывода информации на перфокарты

Носитель информации	.80-колонные перфо- карты
-------------------------------	------------------------------

Представление информации ГОСТ 10859—64 или
любое с программ-
ным перекодиро-
ванием

Скорость вывода 100 карт в минуту
Тип механизма ПЭМ-80

Устройство ввода — вывода с пишущей машинки (УМП)

Представление информации . . .Первые 92 знака,
ГОСТ 10859—64

Скорость ввода — вывода . . .10 знаков в секунду

Тип механизма Пишущая машинка
«Консул-254»

Устройство печати

Представление информации . . .Первые 78 знаков,
ГОСТ 10859—64

Скорость вывода 400 строк в минуту

Количество знаков в строке . .128

Тип механизма АЦПУ-128/2

Накопитель на магнитной ленте

Максимальное количество лентопро-
тяжных механизмов на одно уст-
ройство управления 8

Количество механизмов для основ-
ного комплекта 6

Емкость одного механизма $5,5 \cdot 10^8$ или $11 \cdot 10^8$
символов

Скорость обмена 30 000 или 60 000 сим-
волов в секунду

Тип лентопротяжного механизма .ОЯ4.036.010 Сп (от
машины М-220)
или НМЛ-67

Устройство подготовки данных на перфокартах

Носитель информации 80-колонные перфо-
карты

Представление информации . . .Первые 78 знаков,
ГОСТ 10859—64

Принцип приостановок позволяет совмещать работу вычислителя и внешних устройств.

Обмен вычислителя с внешними устройствами осуществляется по универсальным каналам связи 8-разрядными символами (7 разрядов — информация, 1 — контроль).

Специальные электронные часы позволяют программе «Диспетчер» следить за решением до четырех рабочих программ одновременно.

Понятие о структуре команд, базировании и индексировании

В ЭЦВМ «Минск-32» применены два структурных вида команд: двухадресный и одноадресный.

Двухадресные команды (рис. 1.10) используются для выполнения большинства арифметических, логических, передач управления и некоторых других операций. Команда состоит из следующих частей:

- код операции (КОП) — семь разрядов (0—6-й);
- индекс-адрес (ИА) — четыре разряда (7—10-й);
- первый относительный адрес (А1) — одиннадцать разрядов (13—23-й);
- второй относительный адрес (А2) — одиннадцать разрядов (26—36-й);
- номер базиса А1 (Б1) — два разряда (11-й и 12-й);
- номер базиса А2 (Б2) — два разряда (24-й и 25-й).

Адреса А1 и А2 указываются относительно своих базисных адресов. Для каждой программы можно использовать до четырех базисных адресов: АБ0, АБ1, АБ2 и

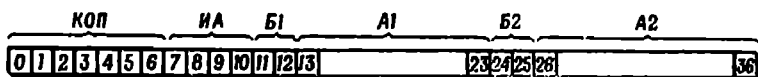


Рис. 1.10. Схема распределения разрядов ячейки памяти при записи двухадресных команд

АБЗ. Для каждого адреса необходимо указывать номер базиса (базисный адрес). Если он не указан, то машина воспринимает базисный адрес как нулевой (АБ0), относительно которого размещается программный массив.

Одиннадцать разрядов относительных адресов (А1 и А2) обеспечивают непрерывную адресацию (нумерацию) внутри массива из 2048 ячеек МОЗУ ($2^{11}=2048$). Для увеличения пределов адресации применяются базисные адреса и индексные ячейки.

Базисные адреса для каждой программы хранятся в двух ячейках соответствующего уровня управляющей области памяти (УОП). Под каждый базисный адрес в этих ячейках отводится по шестнадцать двоичных разрядов. При этом для каждой программы или массива информации доступна любая область МОЗУ емкостью 64К ($2^{16}=64 \times 1024$).

В одноадресных командах (рис. 1.11) в $11 \div 23$ разрядах записывается определитель разновидности данной группы команд, к которой относятся команды:

- работы с индексными ячейками;
- работы с ячейками управляющей области памяти (УОП);
- обращения к внешним устройствам и другие.

Выполнение переадресации (изменения адресных частей команд) может осуществляться с помощью индексных ячеек. Для переадресации А1 и А2 в двухадресной команде используется одна общая индексная ячейка (ИЯ), номер которой указывается в этой же команде (ИА). В индексной ячейке записываются (рис. 1.12) две величины приращения адресов (16-разрядные):

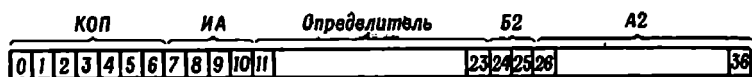


Рис. 1.11. Схема распределения разрядов ячейки памяти при записи одноадресных команд



Рис. 1.12. Схема распределения разрядов индексной ячейки памяти для двухадресных команд

- $\Delta 1$ (разряды 5—20-й) для переадресации $A1$;
 $\Delta 2$ (разряды 21—36-й) для переадресации $A2$.

В переадресуемой команде (в ее ИА) указывается номер (адрес) одной из 15 индексных ячеек (одной группы). Каждая программа и даже участок ее могут использовать свой набор из 15 ИЯ.

Под каждую такую группу выделяется 16 ячеек МОЗУ, но первая ячейка не используется. Место группы ИЯ в МОЗУ определяется программой «Координатор» при загрузке данной программы в МОЗУ. Индексные ячейки отводятся всегда в начале той области МОЗУ, которая выделяется для размещения данной программы. Адрес каждой группы индексных ячеек определяется базисным адресом индекса ($A_{\text{би}}$), который задается с точностью до 16 ячеек. Адрес каждой индексной ячейки ($A_{\text{иа}}$) в группе образуется из базисного адреса индекса ($A_{\text{би}}$) и добавленного к нему 4-разрядного индекса-адреса (номера индексной ячейки), который указывается в команде: $A_{\text{иа}} = A_{\text{би}} + \text{ИА}$.

Например, если $A_{\text{би}} = 123\,460$ и $\text{ИА} = 13$, то адрес индексной ячейки будет равен:

$$A_{\text{иа}} = 123\,460 + 13 = 123\,473.$$

Для модификации содержимого ИЯ, т. е. для изменения величин приращения адресов ($\Delta 1$ и $\Delta 2$), используются индексные константы или константы индексирования (КИ), которые содержат постоянный на данном участке программы шаг переадресации. При записи КИ в индексные ячейки распределение разрядов последних не изменяется (рис. 1.12).

Конечно, модификация содержимого ИЯ выполняется по специальным командам работы с индексами. Например, в системе команд машины имеются команды сложения с индексом (СИ) или вычитания из индекса (ВИ). По этим командам содержимое индексной ячейки может быть увеличено или уменьшено на величину содержимого индексной константы (КИ). Это дает возможность модифицировать адресные части команд как в сторону увеличения, так и в сторону уменьшения.

Для переадресации в одноадресных командах используются те же индексные ячейки, в которых вместо величины приращения $A1$ ($\Delta 1$ в разрядах 5—20-м) или

шага переадресации A1 (при записи индексной константы) записываются нули.

Содержимое индексных ячеек (рис. 1.13) — для команд действий над символами — включает:

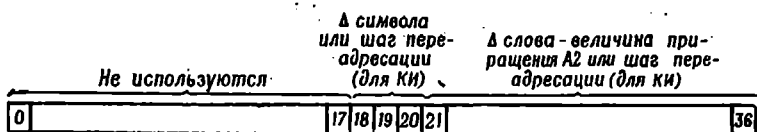


Рис. 1.13. Схема распределения разрядов индексной ячейки памяти для команд обработки символов

— величину приращения A2 ($\Delta 2$ в разрядах 21—36-м), или шаг переадресации A2 (при записи КИ);

— величину приращения адреса символа (разряды 18—20-й), или шаг переадресации номера символа (при записи КИ).

Таким образом, **базированием** называется процесс сложения величины относительного адреса с величиной базисного адреса.

Индексированием называется процесс сложения адресных частей команды с содержимым индексной ячейки, номер (адрес) которой указан в ИА индексируемой команды.

Базирование и индексирование, т. е. формирование исполнительных адресов, выполняются после приема данной команды из МОЗУ в АУ.

Базирование выполняется всегда, а индексирование — тогда, когда индекс-адрес команды не равен нулю ($ИА \neq 0$).

Краткая характеристика системы программирования ЭЦВМ «Минск-32»

ЭЦВМ «Минск-32» имеет развитую систему математического (программного) обеспечения (СМО или СПО), которая состоит из **операционной системы (ОС)** и **системы программирования**.

Операционная система обеспечивает прохождение рабочих программ через машину в процессе их выполнения.

Операционная система рассматривается в главе 4.

Система программирования обеспечивает подготовку программ для решения задач (обработки информации), т. е. следующие процессы программирования:

- программирование на входном языке;
- трансляцию программ с входного языка на внутренний;
- организацию библиотек программ;
- сборку программ и сегментацию задач;
- загрузку программ в МОЗУ для выполнения;
- отладку рабочих программ;
- корректировку программ на входном языке;
- организацию массивов информации на внешних носителях.

Эта система рассчитана на программирование только на входных языках, одним из которых является язык символического кодирования (ЯСК). Поэтому в состав системы программирования входит транслятор с этого языка (ТСК). Вместе с тем система программирования открыта для использования других языков и трансляторов.

В результате трансляции исходной (на ЯСК) программы получается программа на языке загрузки, т. е. на внутреннем языке системы программирования. При этом отдельные программы оформляются стандартным способом в виде отдельных массивов. Каждая программа получает свое имя для обеспечения поиска программ при сборке с другими программами или поиска для загрузки в МОЗУ при выполнении. Программы на языке загрузки обычно хранятся в библиотеках на МЛ.

Для выполнения отдельные или собранные программы загружаются в МОЗУ под управлением системы «Диспетчер», которая отводит им в памяти определенное место с соответствующей настройкой.

Система программирования ЭЦВМ «Минск-32» использует два вида библиотек-программ: библиотеку общего пользования и личные библиотеки пользователей. Все библиотеки хранятся на МЛ.

Магнитная лента с библиотекой общего пользования называется лентой системы (ЛС). На ней хранятся следующие программы:

- программы системы «Диспетчер»;
- программы транслятора (ТСК);

- программы «Библиотекарь»;
- программы ввода — вывода на различные носители информации;
- программы выполнения некоторых стандартных процедур;
- программы вычисления элементарных функций;
- программы для выполнения корректировки текстов на входном языке и сборки программ;
- программы отладочные и другие.

На магнитных лентах личных библиотек пользователей могут храниться следующие программы:

- программа общего пользования на языке загрузки;
- личная программа на входном языке (исходная программа);
- отдельная или собранная личная программа на языке загрузки;
- массивы исходных данных и т. п.

Для трансляции исходную программу (на входном языке) запрашивают из личной библиотеки пользователя. После трансляции программа на языке загрузки записывается в личную библиотеку на ту же или другую МЛ. При записи на ту же МЛ программе обязательно присваивается новое имя.

Если для программы данной задачи емкость свободной области МОЗУ недостаточна, то производится сегментация задачи, т. е. разделение ее на части. В этом случае при решении задачи в МОЗУ обязательно хранится головной сегмент, с которого начинается решение задачи. Остальные (внутренние) сегменты при необходимости под управлением операторов головного сегмента поочередно вызываются с МЛ, загружаются в МОЗУ и выполняются.

Заметим, что каждый сегмент может быть результатом сборки нескольких самостоятельно оттранслированных программ.

Собранными могут быть программы решения отдельных частей данной задачи и стандартные программы из библиотеки общего пользования (с ЛС). Сборка программ производится на языке загрузки по программе «Сборщик» (СБРЩК), которая хранится на ленте системы (ЛС). Программа — результат сборки называется сегментом. В частном случае он может состоять из одной программы. Следовательно, задача может со-

стоять из одного или нескольких сегментов, а сегмент — из нескольких собранных программ или из одной программы. Отладка программ осуществляется по отладочным программам (ОП-1 или ОП-2) из библиотеки общего пользования (на ЛС); при этом отладочная программа контролирует выполнение отлаживаемой и выдает на печать информацию о ходе и результате выполнения ее. Вид и моменты выдачи информации задаются в отладочных операторах.

Эти операторы можно подготовить на уровне ЯСК для отладки программ, протранслированных с ЯСК. Это обеспечивается содержанием словаря символических обозначений в каждой программе на языке загрузки.

Система программирования обеспечивает корректирование любого исходного текста (массива исходных данных или исходной программы), который представлен стандартно и записан на магнитную ленту. Корректировка выполняется программой «Корректор» (ВКОР) по изменениям, задаваемым в виде операторов корректировки. При этом возможны вставка дополнительных строк текста, замена и удаление строк.

После корректировки производится трансляция каждой исправленной программы. Если изменения вносились после отладки собранной программы, то после повторной трансляции заново выполняется сборка.

В ЭЦВМ «Минск-32» применяется стандартная организация массивов информации на внешних носителях (магнитные ленты, перфоленты и перфокарты). Массивы организуются из следующих видов информации:

- исходных данных;
- справочных данных;
- промежуточных или окончательных результатов;
- программ на входном языке или языке загрузки.

Такая организация массивов позволяет для обмена информацией между вычислителем и внешними устройствами использовать набор стандартных программ из библиотеки общего пользования (на ЛС). Массивы хранятся на носителях информационными блоками и разграничиваются контрольными блоками, в которых записывается справочная информация для отличия массивов друг от друга.

Пример последовательности подготовки и прохождения задачи через машину приведен в главе 4 (рис. 4.7).

1.3.3. Единая система электронных вычислительных машин (ЕС ЭВМ)

Содружество социалистических стран — Народная Республика Болгария, Венгерская Народная Республика, Германская Демократическая Республика, Польская Народная Республика, Союз Советских Социалистических Республик и Чехословацкая Социалистическая Республика — приняло решение разрабатывать и производить Единую систему ЭВМ третьего поколения.

Эта система будет единым рядом машин из семи программно-совместимых моделей, которые предназначены для решения многих классов научно-технических и экономических задач в условиях вычислительных центров (ВЦ) и автоматизированных систем управления (АСУ).

ЕС ЭВМ имеет общий для всех моделей состав внешних (периферийных) устройств для обеспечения:

- ввода и вывода перфокарт;
- ввода и вывода перфолент;
- вывода информации с помощью печатающих устройств;
- ввода, вывода и запоминания информации на магнитных лентах и дисках;
- связи с абонентами-пользователями ЭВМ по телефонно-телеграфным линиям с использованием терминальных пунктов и отображением информации на электронно-лучевых трубках.

Присоединение внешних устройств к ЭВМ производится с помощью стандартного интерфейса для обеспечения пополнения и обновления состава этих устройств.

ЕС ЭВМ построена на интегральных схемах на современном конструктивно-технологическом уровне, что обеспечивает хорошие эксплуатационные показатели и массовость промышленного выпуска моделей для стран социалистического содружества.

Для ЕС ЭВМ создается развитая система программного обеспечения (СПО), в состав которой должны входить операционные системы, прикладные программы и комплекс программ технического обслуживания.

Совместимость программирования, стандартизация конструкций и кодов в ЕС ЭВМ создают условия для развития программного обеспечения в процессе исполь-

зования и распространения моделей ЭВМ этой системы в различных областях применения.

Электронная вычислительная машина ЕС-1020 является одной из моделей ЕС ЭВМ и серийно выпускается в СССР с августа 1972 г.

ЭВМ ЕС-1020 предназначена для применения в ВЦ научных и проектных организаций, промышленных предприятий и государственных учреждений для решения научно-технических и информационно-логических задач многих классов.

Она состоит из функционально оформленных блоков с автономной настройкой каждого из них.

Агрегатный тип конструкции позволяет быстро учитывать конкретные условия эксплуатации и повышать производительность ЭВМ подключением дополнительных блоков.

По запросам пользователей могут комплектоваться различные конфигурации блоков. Ниже приведен состав ЭВМ ЕС-1020 минимальной стандартной конфигурации (табл. 3).

Таблица 3

Минимальная стандартная конфигурация ЭВМ ЕС-1020

Наименование устройства	Шифр	Количество
Процессор с оперативной памятью 64 К байта	ЕС-2020-1	1
Пишущая машинка на базе «Консул-260» с БССК	ЕС-7070	1
Устройство управления для накопителей на магнитной ленте	ЕС-5511	1
Накопитель на магнитной ленте (НМЛ-67А)	ЕС-5010	4
Устройство управления для накопителей на магнитных дисках и барабанах	ЕС-5551	1
Накопитель на сменных магнитных дисках	ЕС-5056	2
Устройство ввода с перфокарт	ЕС-6012	1
Устройство ввода с перфоленты	ЕС-6022	1
Устройство-вывода на перфоленту	ЕС-7022	1
Алфавитно-цифровое печатающее устройство	ЕС-7030	1
Устройство вывода на перфокарты	ЕС-7010	1
Устройство подготовки данных на перфокартах	ЕС-9010	2
Устройство подготовки данных на перфоленте	ЕС-9020	2

Оперативная память ЭВМ ЕС-1020 может быть расширена до 256 К байтов. Состав внешних устройств может быть расширен подключением других устройств, входящих в ЕС ЭВМ.

Основные технические характеристики устройств

Процессор (ЕС-2020) — характеризуется широким применением микропрограммного управления, наличием запоминающих регистров и однобайтового арифметическо-логического устройства (АЛУ), разветвленной системой контроля и гибкой логикой прерываний.

Микропрограммная логика процессора служит для управления и исполнения всех инструкций, операций с пульта, процедур обмена данными с внешними устройствами. Структура микрокоманды разрешает задание и выполнение действий над двумя операндами в АЛУ с одновременным обращением к памяти и формированием адреса следующей микрокоманды.

Количество инструкций процессора — 142.

Время выполнения основных операций в *мксек*:

сложение — вычитание с фиксированной запятой — 20—30;

сложение — вычитание с плавающей запятой — 50—70;

сложение — вычитание при двойных словах — 50—110;

умножение с фиксированной запятой — 350;

умножение с плавающей запятой — 480;

умножение при двойных словах — 1200;

деление с фиксированной запятой — 400;

деление с плавающей запятой — 400;

короткие операции — 20—30.

Основная память:

объем (К байт) — 64—256;

время цикла — 2 *мксек*;

время выборки — 1 *мксек*.

Основная память имеет защиту по записи и чтению.

Каналы:

а) мультиплексный:

количество подканалов — 48—128;

скорость передачи данных в мультиплексном режиме — до 16 К байт/сек;

скорость передачи данных в монопольном режиме — до 100 К байт/сек;

количество подключающихся блоков управления — 8;

б) селекторный:

количество каналов — 2;

скорость передачи (К байт/сек) — 200;

количество подключающихся блоков управления — 8.

Операторский пульт (ЕС-1020) — оборудован пишущим механизмом «Консул-260»:

— количество знаков — 93;

— количество клавишей — 47;

— скорость печати — до 10 знаков в секунду.

Печатающее устройство (типа ЕС-7030):

скорость печати — 650—890 строк в минуту;

набор знаков — 82;

количество знаков в строке — 128.

Устройство ввода перфокарт (типа ЕС-6012):

скорость считывания — 500 карт в минуту;

метод считывания — фотоэлектрический;

объем подающего кармана — 1000 карт;

объем приемного кармана — 1000 карт.

Устройство вывода данных на перфокарты (типа ЕС-7010):

скорость перфорации — 100 карт в минуту;

объем подающего кармана — 700 карт;

объем принимающего кармана — 700 карт;

количество принимающих карманов — 2.

Устройство управления для накопителей на магнитной ленте (типа ЕС-5511):

число подключаемых накопителей — до 8;

режим работы с каналом — монопольный.

Накопитель на магнитной ленте (типа ЕС-5010):

скорость ленты — 2 м/сек;

плотность записи — 32 бит/мм; 8 бит/мм;

скорость обмена данными (плотность 32 бит/мм) — 64 К байт/сек;

количество дорожек — 9;

время старта — до 5,5 мсек;

время остановки — до 5,5 мсек;

время перемотки (длина ленты 732 м) — 6 мин;

магнитная головка — комбинированная запись/счит.;

ширина ленты — 12,7 мм;

кассета — стандартная с защитным кольцом.

Устройство управления для накопителей на магнитных дисках и барабанах (типа ЕС-5551):

число подключаемых накопителей — до 8;

режим работы с каналом — монопольный.

Накопитель со сменными магнитными дисками (типа ЕС-5056):

метод записи — с двойной частотой;

плотность информации:

на дорожке 000 — 30 бит/мм;

на дорожке 202 — 44 бит/мм;

количество дисков в пакете — 6;

количество рабочих поверхностей дисков — 10;

количество дорожек — 200 рабочих и 3 резервные;

скорость передачи данных — 156 К байт/сек;

объем памяти — 7,25 байта;

скорость вращения — 2400 об/мин;

время перехода от дорожки к дорожке — 30 мсек;

среднее время доступа — 150 мсек;

минимальное время поиска цилиндра — 150 мсек.

Устройство ввода перфолент (типа ЕС-6022):

скорость — 1500 строк в секунду;

количество двоичных разрядов (дорожек) на каждом шаге — 5; 6; 7; 8.

Устройство вывода перфолент (типа ЕС-7022):

скорость — 150 строк в секунду;

количество двоичных разрядов (дорожек) на каждом шаге — 5; 6; 7; 8.

Система программного обеспечения (СПО) ЕС ЭВМ

Система программного обеспечения (СПО) ЕС ЭВМ является неотъемлемой частью всего ряда моделей. Она предназначена для обеспечения эффективного функционирования расширенного состава технических средств ЕС ЭВМ, повышения эффективности и расширения области применения моделей ряда.

В состав СПО должны входить операционные системы, прикладные программы, комплекс программ технического обслуживания и документация СПО.

СПО должна обеспечивать:

— работу на всех моделях ЕС ЭВМ;

— высоконадежную и программно-управляемую наладку, проверку и диагностику оборудования для облегчения проверки, обнаружения, локализации и устранения неисправностей, проведения профилактических работ и проверки работоспособности всех программных компонентов;

— функционирование однопроцессорных, многопроцессорных и многомашинных вычислительных систем;

— высокий уровень пропускной способности вычислительных систем;

— функционирование ЕС ЭВМ в широком диапазоне времени реакций вычислительных систем;

— автоматическое разделение машинного времени;

— автоматизированный сбор исходных данных непосредственно от источника, их хранение, обновление и обработку в сочетании с автоматизированным составлением, обновлением и выдачей документов непосредственно на месте их использования;

— дистанционный ввод — вывод информации по открытым и закрытым телефонным и телеграфным каналам связи;

— обработку и воспроизведение информации в графической форме на средствах отображения;

— возможность подбора программных средств, соответствующих используемому составу технических средств к решаемым задачам (генерация);

— повышение производительности труда программистов с помощью набора средств автоматизации подготовки (средств) задач к решению на ЭВМ;

— программную совместимость моделей ЕС ЭВМ;

— упрощение процедур по расширению или замене исполняемых функций при изменении состава технических средств или решаемых задач;

— возможность выпуска национальных версий программных компонентов.

Операционные системы СПО ЕС ЭВМ

Операционные системы должны обеспечить единое функционирование моделей ряда, автоматизацию процесса подготовки и эксплуатации программ, а также повысить производительность труда программистов-операторов и обслуживающего персонала. Это достигается:

— увеличением пропускной способности вычислительной системы;

— уменьшением времени реакции системы;

— повышением производительности труда и удобства использования вычислительной системы;

— обеспечением упорядоченного развития системы при расширении областей применения.

Операционные системы имеют следующий состав:

— дисковая операционная система (ДОС/ЕС);

— операционная система (ОС/ЕС);

— малая операционная система для модели Р-20А (МОС/ЕС);

— базовая операционная система для модели Р-10 (БОС/ЕС);

— комплекс программ операционных систем.

Комплекс программ в свой состав включает:

— управляющие программы;

— обрабатывающие программы;

— средства восстановления системы;

— средства генерации системы.

Управляющие программы должны обеспечивать:

— эксплуатацию задач в однопрограммном и мультипрограммном режимах;

— совмещение работы оператора по подготовке очередного задания с обработкой введенных заданий;

— эффективное использование ресурсов системы;

— максимальное сокращение основных функций оператора;

— решение задач в режиме разделения времени;

— решение задач в реальном масштабе времени;

— решение задач в многопроцессорных конфигурациях;

— решение задач по распараллеленному алгоритму;

— решение задач на нескольких ЭВМ с использованием общих полей внешних запоминающих устройств;

— планирование и управление операциями ввода — вывода для системных устройств ЕС ЭВМ;

— автоматический сбор и обработку статистических данных с процессора внутри системы;

— автоматизированное разграничение доступа к хранимой и обрабатываемой информации для прикрепленных абонентов и обслуживающего персонала;

— автоматизированное разграничение доступа к за-

щищенным массивам информации и регистрацию обращений к ним.

Таким образом, управляющие программы осуществляют первоначальную загрузку оперативной памяти и управление всей работой системы, включая обработку прерываний, распределение работы каналов, загрузку программ из библиотеки стандартных подпрограмм в оперативную память. Управляющие программы обеспечивают мультипрограммную работу, осуществляют связь с оператором и представляют пользователю большие возможности для управления массивами информации.

Обрабатывающие программы должны иметь средства облегчения работы программиста и оператора при составлении, отладке и эксплуатации программ на ЭВМ ЕС. Эти программы должны обеспечивать:

- эффективную систему программирования на машинно-ориентированном языке (АССЕМБЛЕР) со средствами отладки на этом языке;

- эффективную систему программирования, которая содержит набор трансляторов для проблемно-ориентированных языков программирования (АЛГОЛ, ФОРТРАН, ПЛ/1, КОБОЛ), библиотеки стандартных подпрограмм и средства отладки;

- системы программирования в диалоговом режиме;

- преобразование программных модулей, представленных на проблемно-ориентированных языках, в объектную форму в целях объединения в выполняемую программу;

- хранение и обновление программных библиотек;

- копирование, перемещение с редактированием, ввод и вывод информации;

- удобство при создании программ для составления печатных отчетов на основе информации, хранящейся в системе;

- сортировку и объединение массивов информации на магнитных лентах и дисках.

Следовательно, обрабатывающие программы операционных систем осуществляют:

- объединение отдельно транслируемых модулей в одну или несколько выполняемых программ;

- составление перекрывающихся программных фаз;

- резервирование памяти для общих областей;

- работу с библиотеками программ с реализацией

функций копирования, обновления, сжатия, выполнения библиотек и создания частных библиотек.

Средства восстановления системы предназначены для всех моделей ЕС ЭВМ и при возникновении ошибки в том или ином устройстве должны обеспечивать:

- регистрацию состояния оборудования;
- анализ неисправности;
- организацию завершения эксплуатации программы решения задачи;
- подготовку информации для процедур устранения ошибки;
- проверку и диагностику внешних устройств в мультипрограммном режиме;
- исправление повреждений в программе;
- повторение прерванной команды центрального процессора или команды канала;
- использование альтернативных путей к устройствам;
- динамическую реконфигурацию.

Средства генерации системы предназначены для обеспечения изменения объема и состава СПО в зависимости от используемого состава технических средств, частей системы программного обеспечения, объема и класса решаемых задач.

Прикладные программы СПО ЕС ЭВМ

Эти программы должны функционировать под управлением операционных систем СПО и включать в свой состав следующие группы:

- первая группа — пакеты прикладных программ, расширяющих возможности операционных систем для специальных комплексов технических средств и способов применения;
- вторая группа — пакеты прикладных программ общего назначения, т. е. единые комплексы программ для решения типовых научно-технических, инженерно-экономических и специальных задач.

Состав первой группы:

- средства для обработки информации на расстоянии;
- средства машинной графики;
- средства для обеспечения специфических внешних устройств;

пакеты для многомашинных комплексов;
пакеты для обработки заданий на расстоянии;
системы программирования для оперативного решения задач;
средства для обеспечения преемственности программ;
интегральный эмулятор ДОС/ОС;
системы программирования на базе проблемно-ориентированных языков;
средства для обмена информацией с машинами-спутниками;

обобщение системы обработки информации.

Дадим краткую характеристику каждой подгруппе.

Средства для обработки информации на расстоянии предназначены для осуществления приема, обработки и передачи данных через телеграфные и телефонные линии связи и обеспечивают:

— построение систем сбора данных от удаленных терминалов с последующей обработкой в центральной системе обработки данных;

— построение систем переключения сообщений, в которых сообщения, получаемые центральной системой обработки данных от одного из удаленных терминалов, передаются в один или более других удаленных терминалов;

— построение справочно-информационных систем, в которых запросы и записи информации получают от нескольких удаленных терминалов и используются для опроса или обновления архивов данных, обслуживаемых центральной системой обработки данных.

Средства машинной графики предназначены для осуществления обработки и воспроизведения информации в графической форме на средствах отображения и обеспечивают:

— создание программ, которые отображают информацию на графическом дисплее (при использовании языка ФОРТРАН IV);

— получение информации из памяти ЭВМ, отображение ее на экранах алфавитно-цифровых и графических дисплеев, проверки, модификации на экранах и возврата информации в память ЭВМ;

— графический анализ трехмерных данных;

— создание систем машинной графики на основе проблемно-ориентированных языков пользователей;

— возможность определения заданий, выполняемых системой, для пользователей графических дисплеев;

— определение задания центральной системе обработки данных на графическом дисплее, входящем в состав удаленного терминала.

Средства для обеспечения специфических внешних устройств предназначены для обеспечения использования следующих устройств:

— ввода — вывода с перфоленты;

— вывода на графопостроители барабанного и планшетного типа;

— «речевого» ввода — вывода информации;

— оптического чтения информации с листа;

— ввода — вывода информации с микрофильмов и микрофиш;

— ввода — вывода информации на аналого-цифровые преобразователи.

Пакеты программ для многомашинных комплексов предназначены для обеспечения функционирования комплекса из нескольких ЭВМ, соединенных через адаптер «канал — канал», внешнее поле памяти ЭВМ и линии связи.

Пакеты программ для вычислений на расстоянии предназначены для обеспечения ввода задания с удаленного терминала, его выполнения на центральной вычислительной системе в режиме пакетной обработки и вывода результатов выполнения задания на удаленный терминал или на локальные устройства. Пользователь с удаленным терминалом имеет возможность подготовить задание заранее или определить его в диалоге с центральной вычислительной системой.

Системы программирования для оперативного решения задач имеют средства для одновременного использования центральной вычислительной системы при решении задач несколькими абонентами и обеспечивают:

— диалоговую обработку заданий под контролем для научно-технических и информационно-логических применений;

— использование ЭВМ в диалоговом режиме с помощью специальных языков программирования.

Средства для обеспечения преемственности программ предназначены для осуществления преемственности про-

грамм на языке ЭВМ тех моделей, которые не входят в ЕС ЭВМ, и обеспечивают:

- конверсию данных, представленных на перфокартах и магнитных лентах, в форматы магнитных лент ЕС ЭВМ;

- преобразование программ, представленных в автокодах, в язык АССЕМБЛЕРА;

- преобразование программ с языков высокого уровня в ПЛ/1;

- интегральную эмуляцию этих ЭВМ на базе аппаратных возможностей моделей ЕС ЭВМ;

- трансляцию программ, представленных на АЛМО.

Интегральный эмулятор ДОС/ОС — это средство пользователя операционной системы ОС/ЕС при использовании программ, написанных для операционной системы ДОС/ЕС.

Системы программирования, созданные на базе проблемно-ориентированных языков обеспечивают выполнение следующих функций:

- решение задач моделирования;

- решение задач логической обработки информации с помощью таблиц решений;

- решение задач обработки символов;

- манипуляции и вычисления матриц;

- автоматическую печать блок-схем для программ, представленных в языке блок-схем;

- автоматическое составление блок-схем для программ, представленных в языках программирования ЕС ЭВМ;

- возможность создания системы программирования на основе собственных проблемно-ориентированных языков пользователя;

- решение задач сортировки и объединения нескольких файлов данных.

Средства для обмена информацией с машинами-спутниками предназначены для обеспечения гибкости и эффективности применения малых ЭВМ, при этом используются вычислительные мощности и память больших моделей ЕС ЭВМ с программами, составленными на языке малых ЭВМ-спутников.

Обобщенные системы обработки информации предназначены для создания, изменения и расширения автоматизированных систем обработки данных и даже АСУ.

Состав второй группы:

пакеты программ для научно-технических расчетов;
система математического программирования;
пакеты программ для моделирования;
системы для обработки документов и текстовой информации;

пакеты программ для планирования и управления;
обобщенные информационные системы.

Назначение каждой подгруппы:

пакеты программ для научно-технических расчетов предназначены для обеспечения пользователя основными стандартными подпрограммами численного анализа и статистики;

система математического программирования включает средства для пользователя, которые обеспечивают решение задач линейного и выпуклого программирования (планирования);

пакеты программ для моделирования предназначены для обеспечения пользователя общецелевым средством моделирования и исследования дискретных и непрерывных процессов в различных системах;

системы для обработки документов и текстовой информации предназначены для создания документационных файлов, которые хранятся в машинной форме и содержат полные тексты аннотации или ключевые слова, при этом система ведения документов автоматически регистрирует документы, выдаваемые из ЭВМ, с указанием пользователя и характеристик документа;

пакеты программ для планирования и управления обеспечивают:

- анализ критического пути и общей стоимости;
- свойства систем ПЕРТ и ПЕРТ-стоимость;
- распределение ресурсов;
- составление отчетов;

обобщенные информационные системы предназначены для обеспечения опроса и обслуживания информационных файлов различной структуры.

Комплекс программ технического обслуживания

Программы этого комплекса предназначены для наладки, проверки и диагностики комплектов технических

средств ЕС ЭВМ. В состав этого программного комплекса входят комплекты тест-программ для технических средств ЕС ЭВМ.

Система документации СПО ЕС ЭВМ

Эта система предназначена для обеспечения эффективного применения и эксплуатации программных и технических средств ЕС ЭВМ и состоит из комплектов документов на программные части СПО ЕС ЭВМ.

1.4. ОРГАНИЗАЦИЯ ОБРАБОТКИ ВОЕННОЙ ИНФОРМАЦИИ В ТИПОВОМ ВОЕННОМ ВЫЧИСЛИТЕЛЬНОМ ЦЕНТРЕ (ТВВЦ)

1.4.1 Назначение, функции и структура ТВВЦ

Различные этапы получения и обработки военной информации (ОТИ) в системе управления войсками характеризуются большой сложностью и многоступенчатостью и требуют применения различных средств техники управления.

Сначала решение этой проблемы шло по пути широкого внедрения различных вычислительных машин в органы управления войсками для механизации (частичной автоматизации) основных этапов управленческой работы.

По мере увеличения применения вычислительных машин в процессе управления войсками очень большое значение для повышения эффективности управления приобретает выбор рациональных форм автоматизации (механизации). Наш опыт и опыт американской армии показал, что наиболее прогрессивным способом применения средств техники управления войсками является их комплексное использование.

Например, эта тенденция проявляется в агрегировании вычислительных машин в технически связанную систему обработки информации. Для агрегирования вычислительных машин характерно использование общего носителя информации, различных преобразователей и объединение вычислительных машин в агрегаты для непрерывного выполнения обработки информации.

Высшей формой агрегирования средств техники управления являются автоматизированные системы управ-

ления (АСУ), так как они охватывают весь процесс обработки информации — от ее получения до использования полученных результатов.

Комплексное использование вычислительных машин выдвигает ряд организационных проблем, решаемых в современных условиях в рамках единой организационной структуры — вычислительного центра (ВЦ).

Вычислительный центр предназначен для систематической автоматизации процесса обработки информации в той системе управления, которую он обслуживает. Глубокий анализ принципов обработки информации в ВЦ, функций и структуры ВЦ дал в своем отчете о поездке в США член-корреспондент АН СССР Ершов А. П. [8].

Многие принципиальные вопросы работы ВЦ были подняты в докладе Ершова А. П. на второй Всесоюзной конференции программистов (ВКП-2, февраль 1970 г., Новосибирск) и на Всесоюзном совещании по применению вычислительной техники и АСУ, которое состоялось в Москве в конце января 1972 г.

Можно выделить следующие основные принципы обработки информации в ВЦ:

единство систематической приемки и подготовки информации к обработке в целях обеспечения однозначности семантики и простоты формирования сообщений;

единый технологический процесс обработки информации — для определения очередности системной (пакетной) обработки массивов информации, выбора способов хранения массивов, выбора алгоритмов обработки массивов и формирования выходных документов;

согласование по времени и функциям обработки информации всех средств управления ВЦ — объединением в технический комплекс;

систематическая выдача обработанной информации в необходимой форме для данной системы управления.

В системе управления войсками вычислительные центры выполняют роль основных органов управления по переработке военной информации.

По взглядам американских военных специалистов, они являются опорными пунктами автоматизированных систем управления различных структур войск и связываются в единую национальную сеть вычислительных центров.

Организационная структура, тип (класс) и штатное

расписание определяются спецификой конкретного ВЦ, т. е. уровнем управления войсками, обслуживаемого данным ВЦ.

Рассмотрим основные функции и организационную структуру обобщенного вычислительного центра (ТВВЦ).

В соответствии с технологическим процессом обработки военной информации (прохождения задач) ТВВЦ выполняет следующие основные функции:

- организацию подготовки информации;
- эксплуатацию и хранение машинных носителей информации;
- обработку информации на ЭЦВМ;
- приемку программ;
- ведение архива;
- учет работы ЭЦВМ;
- эксплуатацию и обслуживание СКМ, СПМ, ЭЦВМ, внешних устройств и другой техники управления.

Для реализации указанных функций в организационной структуре ТВВЦ, как и любого другого ВЦ, должны быть следующие отделы со своими группами (рис. 1.14):

- отдел программирования (проектирования);
- производственный отдел;
- отдел технического обслуживания;
- административно-хозяйственные службы.

Надо иметь в виду, что численность сотрудников каждого функционального подразделения ТВВЦ и взаимозависимость функций персонала определяются объемом выполняемых работ, принятыми нормативами и фактической производительностью труда.

Организация подготовки информации включает следующие обязательные стадии:

- контроль заполнения документов;
- первая перфорация;
- дубль-перфорация;
- сверка результатов первой перфорации и дубль-перфорации;
- логический контроль готового носителя информации;
- подготовка копий готового носителя;
- установка в стеллаж архива.

При проведении контроля необходимо следить за разборчивостью и полнотой заполнения документов черными чернилами или на машинке и наличием всех офи-

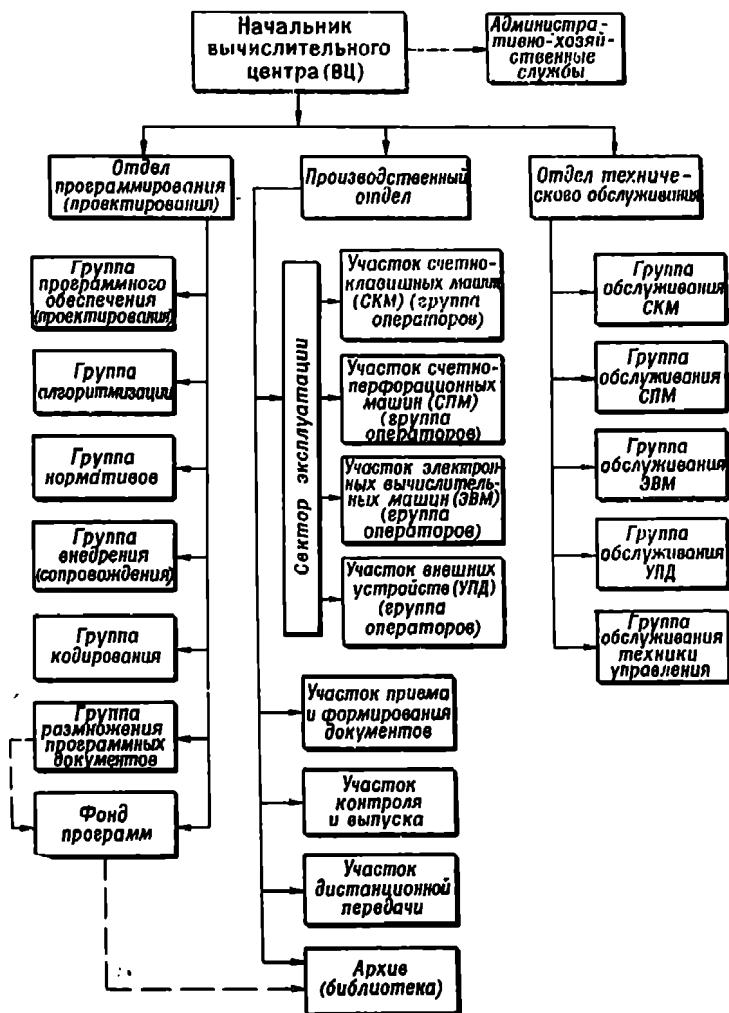


Рис. 1.14. Структурная схема ТВВЦ (по американским взглядам)

циальных подписей на них. Результаты контроля и выяснения сомнительных случаев должны записываться в листки контроля, которые хранятся в архиве ВЦ.

Перфорация должна производиться по единой инструкции и макету.

При подготовке перфоленты после каждого листа надо оставлять участок ее с символом «пусто». В одной

бобине перфоленты должна быть нанесена информация из одной пачки документов. При этом обязательно на бобине должны быть: шифр документа, номер пачки, номер перфорации, дата и подпись исполнителя. После нанесения информации на перфокарты они должны нумероваться.

Дубль-перфорация выполняется по тем же правилам. Сверка результатов двойной перфорации осуществляется на контрольно-считывающем устройстве (КСУ).

Логический контроль реализуется на ЭЦВМ и охватывает формат машинных документов (записей) и попадание показателей в заданный диапазон признаков. Используется стандартная подпрограмма с распечаткой ошибочных записей. Затем носитель при необходимости исправляется и считается готовым, если на одной бобине перфоленты не больше двух склеек. На перфоленту записывается контрольная сумма и лента передается на копирование.

После копирования должно быть не меньше двух экземпляров перфоносителей с контрольной суммой и проверкой на ЭЦВМ. Копии маркируются, сдаются в архив и устанавливаются в стеллажи. В архив передаются и первичные документы.

Все стадии подготовки информации должны документироваться в специальном журнале или карточках.

2. Эксплуатация и хранение машинных носителей. Все машинные носители информации (перфокарты, перфоленты и магнитные ленты) должны храниться в архиве на специальных стеллажах.

На машинных носителях должна быть следующая маркировка:

- регистрационный номер;
- дата подготовки;
- ответственные за подготовку;
- категория (подлинник, первая копия и т. п.);
- регистрационный номер и категория копии;
- количество массивов (только для ПЛ);
- количество вводов (только для ПЛ и ПК);
- контрольная сумма (только для ПК и ПЛ);
- количество склеек (только для ПЛ);
- номер лентопротяжного механизма, на котором размечена данная МЛ;

— номер эксплуатационной карточки (только для МЛ);

— номер информационной карточки (только для МЛ).

В эксплуатационной карточке МЛ указываются:

— дата выпуска магнитной ленты;

— марка и класс ленты;

— количество склеек;

— дата начала эксплуатации ленты;

— даты разметок ленты и замечания по качеству;

— размер ленты в зонах и сведения об изменениях.

Информационная карточка МЛ содержит перечень всех операций по записи массивов на данную ленту с указанием наименования и ссылки на формат массивов, адрес хранения массива, задачи, лица, использующего массив, даты записи и фамилии лица, производившего запись.

При уничтожении некоторого массива на ленте он должен вычеркиваться из информационной карточки.

Обработка информации на ЭЦВМ. Для организации процесса обработки информации на ЭЦВМ можно рекомендовать следующие общие правила:

разделение комплекса программ на этапы с возможностью повторения их;

запись результатов обработки информации на каждом этапе на МЛ;

использование в комплексе управляющей программы диспетчера;

запись больших массивов нормативно-справочной информации на отдельные МЛ;

контроль на всех этапах обработки информации;

различные формы вывода результатов обработки информации, предусматриваемые в программе обработки.

Перед выполнением обработки информации на ЭЦВМ необходимо скомплектовать исходную информацию, т. е. подобрать и проверить готовность всех видов информации для данной обработки.

Часто при машинной обработке информации используется нормативная и другая относительно постоянная информация. Поэтому целесообразно в ВЦ создавать на перфокартах, МЛ и других носителях картотеки нормативно-справочной информации. Эти картотеки необходимо систематически обновлять при изменении обслуживаемой системы управления. Скомплектованная ин-

формация различных видов передается на участок ЭВМ вычислительного центра и сопровождается выдачей описания программы и инструкции по обработке информации.

Напомним, что проверка работы ЭЦВМ осуществляется ежедневной профилактикой с помощью тест-программ. Процесс обработки информации сопровождается программным и аппаратным контролем. Обработка информации на ЭЦВМ по задачам (прогон задач) производится группой операторов ЭЦВМ производственного отдела по недельному графику. Перед эксплуатацией задачи на ЭЦВМ оператор обязан по инструкции подготовить и собрать все необходимые машинные носители информации, сделав отметку об изъятии их из стеллажей архива.

Рекомендуется по эксплуатационной карточке МЛ или журналу эксплуатации задачи проверить качество функционирования МЛ.

Эксплуатация задачи осуществляется по инструкции. При этом ведется журнал эксплуатации задачи, в котором оператор указывает дату, свою фамилию, время выполнения работы; регистрирует различные сбои оборудования и МЛ, ошибки в подготовке данных и свои, недостатки обслуживания и в инструкции и т. д.

После окончания обработки информации на ЭЦВМ оператор должен сделать запись в журнале учета работы машины, а все носители установить на свои места в архиве с записью в соответствующий журнал.

Порядок приемки программ. Выделим следующие этапы приемки:

- проверка программной документации;
- решение тестовой (контрольной) задачи;
- обучение персонала, эксплуатирующего данную задачу;
- опытная эксплуатация задачи (сдающими и принимающими);
- составление акта результатов опытной эксплуатации;
- приемка программ после корректировки.

В программной документации обязательно должны быть отражены следующие вопросы:

перечень использованных символов, шифров и сокращений;

содержание и форма вывода информации;
алгоритм обработки информации (решения задачи);
перечень используемых стандартных программ;
необходимые инструкции (по подготовке носителей,
формированию массивов, ведению нормативов и обра-
ботке информации);

наличие принципиальных блок-схем;
принципы организации и ведения массивов информа-
ции;

разделение комплекса программ на этапы с указан-
ным сопровождением и т. д.

Окончательная программная документация (после
корректировки) оформляется в двух экземплярах, один
из которых сдается в архив.

Приемка программ осуществляется представителями
группы операторов ЭЦВМ из производственного отдела
и группы сопровождения из отдела программирования
(проектирования).

Ведение архива. Архив и библиотека должны вклю-
чать следующие разделы:

документация по системам математического (про-
граммного) обеспечения;

документация по рабочим программам;

карточки отладки программ;

эксплуатационные и информационные карточки МЛ;
журналы (использования ЭЦВМ; эксплуатации за-
дач; приемки, перфорации и выдачи документов);

рабочие носители информации;

листки контроля документов с каталогом;

архивная исходная и результирующая информация с
каталогом;

бланки всевозможных журналов и карточек;

информационные, справочные и печатные материалы;

эксплуатационные документы ЭЦВМ и всех техни-
ческих средств ВЦ;

документы о поступлении, списании материалов и
оборудования и др.

Необходимо выделять следующие виды носителей ин-
формации:

архивные носители систем математического обеспе-
чения;

рабочие носители систем математического обеспе-
чения;

носители рабочих программ;
носители постоянной информации (нормативов);
носители оперативной информации;
носители промежуточных, окончательных результатов
и накопленной информации.

Время хранения различных видов документов архива и библиотеки, требования к оборудованию и условиям для помещений архива изложены в соответствующих руководствах [29].

Учет работы ЭЦВМ. Все время работы ЭЦВМ (машинное время) должно учитываться до одной минуты в специальном журнале. Запись в журнал делает оператор, работающий в данное время, или дежурный инженер.

При учете машинного времени рекомендуется использовать официальную классификацию.

Общее время работы ЭЦВМ подразделяется на следующие виды:

- производительное время;
- непроизводительное время;
- неиспользуемое время (ЭЦВМ отключена);
- время внешних задержек (неисправна сеть питания и т. п.);
- время отсутствия работы;
- неопределенные потери времени (спорные потери).

Тщательный учет расхода машинного времени обеспечивает накопление статистических данных для анализа качества работы ЭЦВМ, приемов ее обслуживания и использования.

Эксплуатация вычислительной техники и оборудования ВЦ. Порядок и последовательность обслуживания оборудования регламентируются технической документацией на отдельные образцы вычислительной техники и оборудования. Для обеспечения выполнения основных функций ТВВЦ в технологическом процессе обработки информации целесообразно кроме комплектов вычислительных машин и устройств подготовки данных в состав вычислительного центра включить следующее оборудование:

- копировально-множительные устройства;
- устройства для перемотки перфоленты;

— стеллажи и шкафы для хранения носителей информации (МЛ, ПЛ и ПК).

Для увеличения эффективности использования вычислительных машин в ВЦ при необходимости рекомендуется модифицировать или «доработать» их. Это зависит от конкретных условий работы и типов применяемых машин. Необходимо, например, поддерживать совместимость лентопротяжных механизмов всех ЭЦВМ.

1.4.2. Функции основных подразделений ТВВЦ по технологическому процессу обработки информации

Основные функции по технологическому процессу обработки информации большинства подразделений ТВВЦ (рис. 1.15) не зависят от типа ЭЦВМ и другого оборудования вычислительного центра, а определяются целью и задачами управления той системы управления войсками, которую обслуживает данный вычислительный центр.

Охарактеризуем основные функции групп отдела программирования (проектирования).

Группа программного (математического) обеспечения проектирует и внедряет в эксплуатацию различные системы программного (математического) обеспечения (трансляторы с алгоритмических языков, интерпретирующие и операционные системы, стандартные и служебные программы и т. п.), разрабатывает собственные программы и обучает использованию системы. Кроме того, она создает и эксплуатирует операционные системы для программистов своего отдела, для групп производственного отдела и группы обслуживания ЭЦВМ отдела технического обслуживания.

Группа алгоритмизации разрабатывает машинные алгоритмы новых задач, крупных изменений эксплуатируемых задач и выдает задания группе кодирования.

Группа кодирования составляет и отлаживает программы, разрабатывает документацию к ним и помогает группе операторов сектора эксплуатации производственного отдела в эксплуатации задач.

Группа сопровождения осуществляет приемку, испытания, эксплуатацию и изменения своих и «сторонних» задач.

Группа нормативов осуществляет функции по созданию, хранению и поддержанию в достоверном состоянии

картотек постоянной информации и нормативно-справочных документов.

Группа размножения документов размножает в необходимом количестве выходные документы.

Отдел программирования выдает заказы:

отделу технического обслуживания — на подготовку МЛ;

производственному отделу — на перфорацию программ и отладку.

Порядок комплектования фонда программ.

Все новые программы после отладки на ЭЦВМ проходят испытания (опытную прогонку) и включаются в фонд программ ВЦ.

Если программа составляется из фондовых программ, то и она после испытания включается в фонд программ (как самостоятельная).

Если комплекс программ состоит из фондовых программ, объединенных только записью на МЛ и инструкцией оператору, то этот комплекс проходит испытания и включается в фонд программ (как самостоятельная программа).

Если программа разработана в другой организации, то ее использование в различных программах или их комплексах возможно после включения в фонд программ.

Для включения программы в фонд необходимы следующие документы:

- паспорт программы;
- текст программы;
- текст оригинальной части на носителе информации;
- контрольный пример для просчета с массивами информации;
- инструкция по вводу программы в эксплуатацию и технологии ее обслуживания;
- инструкция (для оператора) по текущей эксплуатации программы;
- акт производственного испытания программы;
- информационный листок;
- информационная карта.

Содержание и требования к указанным документам определены руководящими материалами Государствен-

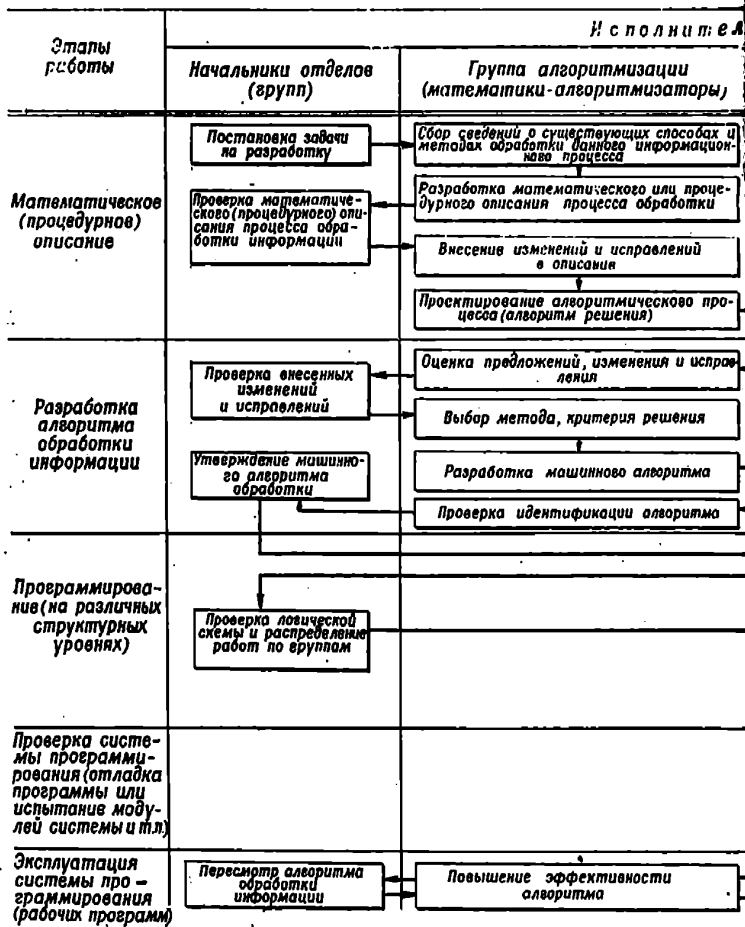
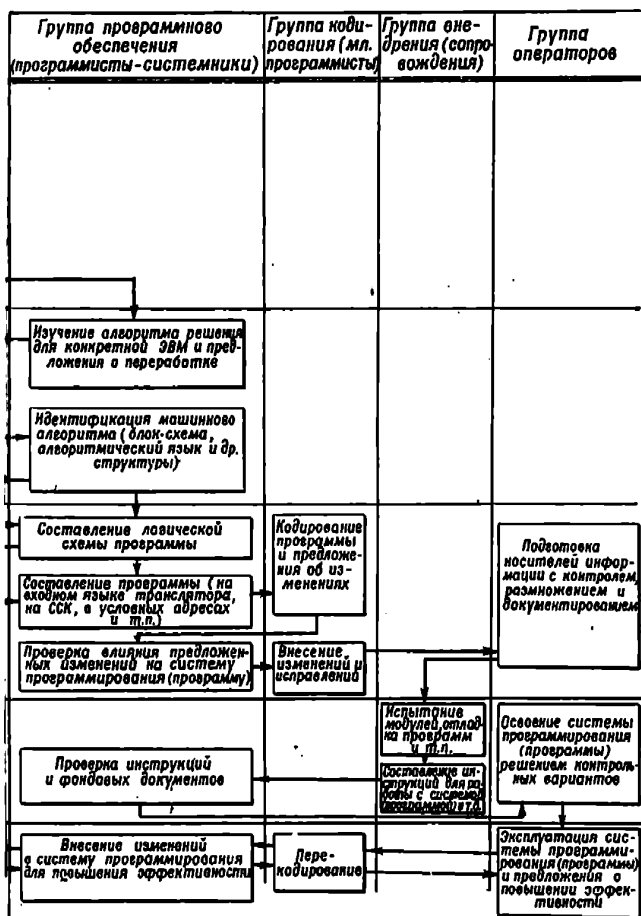


Рис. 1.15. Обобщенная схема технологического

ного комитета по науке и технике при Совете Министров СССР [29].

Показатели качества работы основных подразделений ТВВЦ.

Нельзя рекомендовать единую комплексную оценку качества работы всего ВЦ. Целесообразно применять оценки качества работы отдельных подразделений на



процесса «прохождения» задач в ТВВЦ (вариант)

важнейших этапах технологического процесса обработки информации.

Для отдела программирования оценкой качества работы является суммарный объем составленных, отлаженных программ в пересчете на одного сотрудника отдела в работе за это время;

для группы нормативов — количество ошибок за от-

четный период при обработке на ЭЦВМ в пересчете на 1 млн. знаков нормативно-справочной информации;

для группы операторов устройств подготовки данных — количество ошибок на носителях в пересчете на 1 млн. знаков и количество подготовленных данных на одного работающего.

1.4.3. Методика определения количества ЭЦВМ для ТВВЦ

Опыт работы ВЦ американской армии показывает, что основанием для определения типа и состава вычислительных средств и оборудования являются:

- информационные и временные характеристики задач, которые решаются в интересах обслуживаемой системы управления войсками;

- характеристики технологического процесса обработки информации в ВЦ;

- эксплуатационные характеристики вычислительных средств (машин).

Информационные и временные (основные) характеристики задач:

- носители входной и выходной информации;

- объемы входной и выходной информации по этим носителям;

- объемы и сроки выполнения работ по решению задач;

- информационные форматы, используемые при решении задач;

- формы и способы представления результатов решения задач потребителям.

Характеристики технологического процесса обработки информации:

- типы технологических операций, выполняемых при решении задач;

- вероятностные оценки количества технологических операций каждого типа.

Необходимо также учитывать назначение, состав и следующие эксплуатационные характеристики вычислительных средств:

- производительность при выполнении технологических операций;

- надежность работы устройств и блоков вычислительных машин («автономно» и комплексно);

- совместимость во времени выполнения отдельных технологических операций;
- совместимость работы средств различных типов;
- состав и количество обслуживающего персонала;
- площадь — для размещения комплекса средств;
- стоимость комплекса.

После выбора типов ЭЦВМ определяется их количество. Методика по определению количества ЭЦВМ предусматривает однотипность машин в ВЦ и работу их в режиме мультипрограммирования (многопрограммной обработки).

Оценка необходимого количества ЭЦВМ основана на определении необходимого количества машинных операций для решения задач при максимальной суточной нагрузке машин.

При определении количества ЭЦВМ для укомплектования ВЦ может быть использована формула

$$N_{\text{ЭВМ}} = \frac{K_a \sum_{i=1}^n C_i Q_i}{v_{\text{ср ЭЦВМ}} K_{\text{эфв}} K_{\text{н}} T_p},$$

где K_a — коэффициент времени на автоматическое программирование и выполнение служебных программ режима мультипрограммирования;

C_i — удельное количество машинных операций на решение i -й группы задач ($i=1, 2, \dots, n$) для обработки одного показателя из МОЗУ машины;

Q_i — максимальный суточный объем информации в показателях i -й группы задач;

$v_{\text{ср ЭЦВМ}}$ — среднее быстродействие ЭЦВМ выбранного типа;

$K_{\text{эфв}}$ — коэффициент понижения среднего быстродействия ЭЦВМ выбранного типа от обращения к внешним устройствам машины (НМЛ и УВВ);

$K_{\text{н}}$ — коэффициент использования машинного времени ЭЦВМ выбранного типа (с учетом затрат времени на устранение неисправностей и на регламентные работы);

T_p — полное суточное время работы ЭЦВМ (по плану), сек.

Значение величины коэффициента K_a повышается с увеличением уровня автоматизации. При современном уровне это значение принимается равным 1,2 ($K=1,2$). Удельное количество машинных операций C_i зависит от характера решаемых задач. Среднее быстродействие ЭЦВМ определяется количеством операций, выполняемых процессором (вычислителем машины) без обращения к внешним устройствам. Величина этого параметра $v_{\text{эцвм}}$ зависит от состава операций в программах решаемых задач.

Величина коэффициента $K_{\text{эфв}}$ определяется по формуле

$$K_{\text{эфв}} = \frac{\sum_{i=1}^n Q_i t_{i0}}{\sum_{i=1}^n Q_i t_{ic}},$$

где t_{ic} — среднее время выполнения машинных операций при решении i -й группы задач для обработки одного показателя, хранящегося в МОЗУ машины;

t_{i0} — среднее время, которое затрачивает ЭЦВМ на обработку одного показателя при решении i -й группы задач с учетом времени вывода и обращения к внешним запоминающим устройствам машины.

Исходя из опыта работы вычислительных центров общего применения (АН СССР и др.) для ориентировочных расчетов можно сделать следующие рекомендации:

при обработке больших массивов информации для вычислительных процессов значение удельного количества машинных операций на один показатель исходных данных принимается равным 500 ($C_i=500$); а для комплексных процессов $C_i=10^4$;

программы обработки информации по вычислительным методам содержат до 90% операций типа сложения и до 10% типа умножения;

для многопрограммных ЭЦВМ («Урал-16», БЭСМ-6, «Минск-32» и др.) значение коэффициента $K_{\text{эфв}} = 0,8$.

При использовании в ВЦ машин без совмещения работы процессора (вычислителя) с работой внешних

устройств значение $K_{эфв}$ изменяется в зависимости от характера решаемых задач и применяемых устройств ввода и вывода (коэффициент может уменьшаться до 0,2—0,1).

Количество СПМ и СКМ для ВЦ определяется с учетом неравномерности поступления потоков информации на машинную обработку. Обычно исходят из максимального объема работ («пиковые» нагрузки) и ведут расчет по нормам выработки, утвержденным ЦСУ СССР в 1965 г. [29].

Уточнение выбора типа и количества оборудования ТВВЦ производится на основе анализа результатов расчетов и дополнительного обследования системы управления войсками, для обслуживания которой предназначена данный вычислительный центр.

Глава 2

ОБЩИЕ ПРИНЦИПЫ ПОДГОТОВКИ ВОЕННЫХ ЗАДАЧ К РЕШЕНИЮ НА ЭЦВМ

2.1. МЕТОДИКА РАЗРАБОТКИ ЗАДАЧ

2.1.1. Понятие о постановке задачи

Современные ЭЦВМ имеют высокое быстродействие и большую точность вычислений и выполняют разнообразные процессы по обработке информации:

- производят различные вычисления;
- сортируют сведения по определенным признакам;
- выдают необходимые справки и т. д.

Но при использовании ЭЦВМ для автоматизации обработки информации (процессов управления) необходимо учитывать недостатки современных ЭЦВМ;

— большой объем трудоемкой и кропотливой работы по подготовке задач к решению на машине, что связано со значительными сложностями и затратами времени;

— большое количество технической работы по кодированию информации и подготовке ее к вводу в машину.

Следовательно, на ЭЦВМ целесообразно решать такие задачи, которые удовлетворяют двум основным требованиям:

многократности решения данной задачи в процессе управления или эксплуатации системы; выполнение этого требования при частом, и особенно систематическом решении задачи на машине, приводит к временному и экономическому эффекту, несмотря на большие первоначальные затраты средств и времени;

цикличности процесса решения задачи; это требование связано с относительно небольшой емкостью ОЗУ современных ЭЦВМ.

Современная методика решения большинства задач предусматривает выделение из процесса обработки информации таких элементарных участков, которые повторяются много раз.

Естественно, что за счет многократно повторяющихся участков процесса обработки информации можно программу самого процесса сделать компактной и не очень «длинной».

Для решения любой задачи на ЭЦВМ необходимо выполнять определенную подготовительную работу, в которой выделяются следующие этапы:

- разработка постановки (общего описания) задачи с соответствующими обоснованиями;

- математическое описание задачи и выбор метода ее решения;

- построение алгоритма процесса решения;

- программирование (описание алгоритма решения на определенном языке).

В процессе разработки постановки задачи необходимо определить тактико-технические требования, все факторы и условия, от которых зависит решение задачи.

Сущность требований и выбор факторов определяют назначением результатов и требуемой точностью решения.

В общем описании задачи необходимо определить последовательность обработки информации для всех возможных вариантов исходных данных и условий, которые могут возникнуть в процессе решения данной задачи.

Полное общее описание задачи передается математику-алгоритмизатору, который разрабатывает математическое обеспечение решения данной задачи на ЭЦВМ. В частном случае он подбирает соответствующие аналитические зависимости для описания процессов, составляющих содержание этой задачи.

Таким образом, постановка задачи определяет:

- целевую установку, т. е. цель решения на ЭЦВМ данной задачи в определенном звене процесса управления;

- когда, с какой периодичностью и в каких условиях обстановки (например, для оперативно-тактических задач — в подготовительный период боевых действий или с определенной периодичностью в ходе боя и т. п.) должна решаться на ЭЦВМ задача;

— какие факторы и параметры необходимо учесть при решении задачи (информационный объем задачи, группировку своих войск и т. п.);

— основные оценки (критерии) решения, принимаемые ограничения и допущения;

— что и в какой форме необходимо получать в результате решения задачи.

Кроме того, в постановке задачи должна быть указана постоянная и переменная входная информация, т. е. исходные данные для решения задачи [11].

Постоянная информация не изменяется при решении любого количества вариантов данной задачи. Для оперативно-тактических задач (ОТЗ) такой информацией могут быть следующие данные:

— тактико-технические характеристики (ТТХ) вооружения, боевой техники и транспорта;

— характеристики точности доставки ядерных и других зарядов к объектам противника;

— организационно-штатная структура войск;

— оперативно-тактические нормативы и другие данные.

Переменная информация может изменяться при решении каждого варианта данной задачи.

Для ОТЗ к этой информации относятся:

— сведения об обстановке и характере действий войск;

— положение войск (координаты элементов боевого порядка);

— координаты, размеры и защищенность объектов противника;

— характеристики маршрутов движения;

— условия погоды, время года и другие факторы.

Постановка задачи должна заканчиваться определением содержания и формы выводимых из машины результатов решения (на печать, на АЦПУ, на выходной перфоратор, на преобразователи и в каналы связи, на средства наглядного отображения и т. п.).

2.1.2. Математическое описание (модель) задачи

При постановке задачи общая последовательность ее решения формулируется в терминах и понятиях соответствующей области знаний.

ЭЦВМ перерабатывают информацию, производя действия над числами (командами), которые выражают количественные соотношения.

Как же перейти от физического смысла решаемой задачи с учетом всевозможных факторов, условий и качественных оценок к арифметическому решению задачи на ЭЦВМ? Начальный переход осуществляется на этапе разработки математического описания задачи, которое является основной частью математического обеспечения автоматизации решения задач — нового направления прикладной математики.

В математическом описании задачи необходимо очень подробно описывать порядок и последовательность получения результатов, предусмотренных в постановке задачи, с помощью разнообразного математического и логического аппарата. При этом необходимо раскрыть физическую сущность отдельных этапов решения, взаимосвязь между ними и порядок перехода от этапа к этапу решения.

Математический аппарат позволяет количественно учитывать различные факторы, влияющие на результаты решения задачи, и создает условия для составления алгоритма решения задачи, т. е. для подготовки переложения порядка решения задачи на язык ЭЦВМ (программа решения задачи).

Сущностью математического описания задачи является разработка комплекса аналитических зависимостей, которые учитывают заданное число факторов и обеспечивают решение поставленной задачи с необходимой точностью.

Для решения вручную многих задач была разработана соответствующая методика и составлены математические описания. Но для решения этих задач на ЭЦВМ математические описания преобразовываются так, чтобы этапы алгоритмизации и составления программы были менее трудоемкими. Для этого можно рекомендовать усложнение формул, если значительно уменьшается количество логических условий для использования более простых формул при решении вручную.

Естественно, что вид математических зависимостей, их сложность или простота определяются не техническими средствами обработки информации, а содержанием решаемой задачи.

Например, при решении информационно-логических задач (задачи сортировки, учета и т. п.) применяется элементарный математический аппарат (без сложных формул), а ЭЦВМ используется как средство сортировки и обработки информации.

Для решения большинства военных и военно-технических задач математическое описание часто разрабатывается впервые. Это объясняется тем, что подобные задачи до применения ЭЦВМ решались вручную на «качественном уровне», т. е. на основе накопленного опыта (интуиции) людей и наличием достаточно большого количества специфических задач и ситуаций, не повторяющихся в дальнейшем.

Следовательно, для разработки математического описания военных и военно-технических задач необходима такая математическая методика, которая обеспечивает количественный учет всех факторов, выражаемых обычно качественно через опыт. Количественные основы для важнейших факторов разработаны, но общие связи их с учетом морально-политических, экономических и других факторов являются самой большой трудностью для математического описания.

Математическое описание боевых действий — очень сложная задача, решить которую в большинстве случаев можно только методом последовательного приближения.

Реальные боевые действия характеризуются неопределенностью обстановки, так как осуществляются в условиях отсутствия полной информации и противодействия противника. Поэтому изменение большинства факторов учитывается статистически, а при решении задач на ЭЦВМ используются датчики случайных чисел.

Таким образом, при разработке математического описания (математической модели) военных и военно-технических задач можно рекомендовать такую последовательность:

- уяснить сущность материальной основы происходящего процесса;

- выявить основные факторы, от которых зависят результаты решения; при этом необходимо установить характер изменения этих факторов в зависимости от других величин и условий;

- разработать или подобрать математические фор-

мулы, которые отражали бы аналитически зависимость основных факторов от всех других величин и условий;
— подобрать математический аппарат для объединения рассчитываемых величин в зависимости от всех остальных факторов, определяющих результаты решения.

Выполнив работы в указанном объеме, математик-алгоритмизатор получает математическое описание (математическую модель), которое может иметь различную степень сложности с привлечением любого уровня формализации, т. е. любой математической символики.

Математические модели военных и военно-технических систем связаны с переработкой больших объемов информации.

Это вызывается неопределенностью большинства военных и военно-технических задач, т. е. с имеющимся множеством решений первоначально поставленной задачи.

В таких случаях возникает новая, дополнительная задача: из множества всех решений первоначально поставленной задачи выбрать наиболее выгодное.

Естественно, должно быть точно указано, в каком смысле понимается выгодность того или иного решения. Другими словами, выбирается оценка решения или критерий эффективности.

Это значит, что по определенному специальному методу перебираются все возможные решения первоначальной неопределенной задачи и из этих решений выбирается такое, которое наиболее выгодно по выбранному критерию.

Выбор критерия и математического метода для получения оптимального решения производится одновременно с разработкой математического описания задачи.

Критерий — это средство для решения, которое является оценкой моделируемого (описываемого) процесса.

За критерий принимается совокупность количественных характеристик, которые позволяют судить об эффективности выполнения поставленной задачи.

Критерий для получения оптимального решения должен строго соответствовать постановке задачи и иметь ясный физический смысл.

Выбор критерия — один из наиболее сложных вопросов, решаемых при разработке статистических задач вообще и военных, и военно-технических задач в частности.

От правильного выбора критерия во многом зависит эффективность использования сил и средств в боевых действиях. Часто критерий содержит противоречивые требования, выполнить которые полностью очень трудно.

Например, планируется массированный огонь n артиллерийских подразделений по m важным объектам противника.

Критерием оптимального решения этой задачи может быть нанесение максимального поражения объектам противника при минимальной затрате расходуемых боеприпасов

$$\left(\sum_{i=1}^n \sum_{j=1}^m M_{ij} = \max \text{ при } \sum_{i=1}^n N_i = \min \right).$$

Требования, выдвигаемые в этом критерии, противоречивы. Возникает проблема компромиссного выполнения противоречивых требований, содержащихся в критерии, с тем чтобы результат решения был оптимальным.

В соответствии с выбранным критерием для решения задачи требуется такой математический метод, который дал бы оптимальный результат.

При выборе того или иного способа действий войск (решение военных и военно-технических задач) органы управления должны стремиться, чтобы выбранный ими способ действий приводил к наилучшим результатам (выполнение поставленной задачи с минимальной затратой сил и средств, нанесение противнику максимального ущерба имеющимися средствами и т. п.).

Такой способ действий называется оптимальным выбором или оптимальным решением.

Применение того или иного математического метода зависит от сложности условий, наличия информации и необходимости получения в этих условиях оптимального выбора.

Потребность в научном подходе к выбору математического метода решения возрастает с расширением и усложнением задач: чем сложнее задача, тем труднее найти оптимальное решение.

Выбор метода зависит от многих обстоятельств, которые часто трудно совместить.

Разные задачи предъявляют к методам различные требования, т. е. метод может быть хорошим для решения одних задач и непригодным для решения других.

Практические потребности теории решений обусловили появление и развитие не только новых математических методов решения задач, но и новых математических направлений: теории игр, теории массового обслуживания, линейного и динамического программирования, математической теории оптимальных процессов и т. д.

Необходимо четко представлять, что работа на этапе подготовки задачи к решению на ЭЦВМ требует высокой квалификации, является творческой и трудно поддается формализации.

При разработке математического описания (модели) задачи обязательно должен быть выделен весь комплекс логических условий решения и уточнены содержание, объем и диапазон переменных исходных данных, используемых при решении данной задачи.

2.1.3. Понятие об алгоритме решения задачи

При разработке математического описания задачи делается математизация ее, т. е. первый шаг перехода от физического смысла задачи (в терминах соответствующей области знаний) к арифметическому решению на ЭЦВМ. Как уже отмечалось, в общем случае математические модели военных и военно-технических систем связаны с переработкой большого объема информации, поэтому реализация их возможна только с применением ЭЦВМ.

Для этого надо иметь строгую последовательность выполнения соответствующей переработки информации. Другими словами, необходим алгоритм решения задачи.

По определению академика В. М. Глушкова, «однозначные преобразования слов в тех или иных алфавитах, задаваемые с помощью конечных систем правил, принято называть алгоритмами».

Инженер-программист получает от математика-алгоритмизатора математическое описание задачи. Большинство задач вычислительного характера описываются: дифференциальными уравнениями, векторами, тригонометрическими функциями, интегралами и другими терминами, которые ЭЦВМ не может непосредственно переработать.

Для решения подобных задач на ЭЦВМ необходимо выбрать соответствующий численный метод.

Выбор численного метода осуществляется с учетом особенностей машинного счета и обеспечения следующих требований:

цикличности вычислений в процессе решения задачи;
заданной точности решения;

минимальной загрузки памяти машины, что зависит от связности алгоритма.

Пример. Вычислить значение функции $y = \sin x$. Из постановки задачи видно, что математическое описание задачи выражено формулой $y = \sin x$. Необходимо составить алгоритм для вычисления $\sin x$ на ЭЦВМ.

Известно, что $\sin x$ можно разложить в ряд Тейлора и с помощью этого ряда и арифметических операций вычислить значение функции. Допустим, что точность решения ограничена четырьмя членами разложения:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!}.$$

Представим этот многочлен по схеме Горнера:

$$\begin{aligned} & \left(\left(\left(\left(\left(-\frac{1}{7!}x + 0 \right)x + \frac{1}{5!} \right)x + 0 \right)x - \frac{1}{3!} \right)x + 0 \right)x + 1 \right)x = \\ & = \sin x. \end{aligned}$$

Получен алгоритм решения задачи вычисления значения функции, который можно преобразовать в рабочий (решающий) алгоритм и программу решения задачи для конкретной ЭЦВМ.

В общем случае по математическому описанию с учетом переменных исходных данных составляется алгоритм решения задачи, для чего используются и основные законы логики.

Главной целью составления алгоритма (алгоритмизации) является разработка правильной последовательности процесса переработки информации. Следовательно, составление алгоритма решения сводится к разработке последовательности выполнения расчета и выявлению всех логических условий, при которых должна использоваться каждая математическая формула.

Главное в алгоритме — изложение динамики процесса переработки информации, логика процесса, т. е. выявление того, как и в какой последовательности необходимо осуществлять решение задачи в зависимости от конкретных значений исходных данных и промежуточных результатов вычислений.

При разработке алгоритма решения данной задачи на ЭЦВМ, как правило, возникает несколько вариантов алгоритмов, отличающихся по структуре, объему работы, количеству информации, которая возникает в процессе решения задачи. От выбранного (по этим факторам) алгоритма зависит эффективность использования ЭЦВМ для решения данной задачи.

Выбранный алгоритм решения большинства задач необходимо фиксировать в той или иной форме, так как только алгоритмы решения простейших задач можно мысленно представлять.

Обычно алгоритмы решения описываются на каком либо алгоритмическом языке, в качестве которого сейчас широко применяется АЛГОЛ-60.

Для описания на алгоритмическом языке выбранный алгоритм решения расчленяется на ряд самостоятельных частей, которые называются операторами или блоками и схемы решения задачи.

Разработка алгоритма решения требует высокой квалификации от программистов, связана с большими трудностями и отсутствием формальных правил.

При разработке алгоритмов решения задач, в которых исследуемые физические явления мало изучены, очень трудно установить пределы изменения всех величин, участвующих в решении. Но учитывать порядок исходных данных, промежуточных и окончательных результатов необходимо при решении любой задачи на ЭЦВМ.

Если порядок чисел при решении задачи будет превышать диапазон представления чисел в конкретной ЭЦВМ, производится выбор масштабных коэффициентов и преобразование расчетных формул, т. е. масштабирование.

2.1.4. Понятие о программе решения задачи на ЭЦВМ

Содержанием этого этапа работы является построение рабочего алгоритма, равнозначного алгоритму решения задачи и выполняемого на данной ЭЦВМ, т. е. происходит перевод алгоритма решения задачи на язык машины. В результате выполнения работы на этом этапе должна быть получена рабочая программа решения задачи на ЭЦВМ [27].

Необходимо четко понимать разницу между алгорит-

мом решения и рабочим алгоритмом, так называемым решающим алгоритмом.

Это различие заключается в уровне алгоритмов, т. е. в степени подробности описания их.

На более ранней стадии разработки алгоритма решения конкретная ЭЦВМ, на которой будет этот алгоритм реализован, еще не выбирается. Поэтому процесс решения задачи расчленяется на последовательность арифметических и логических операторов, которые составляют схему решения задачи. Эта схема выражает характер процесса решения, но для автоматического решения задачи на ЭЦВМ ее недостаточно.

Необходимо схему решения дополнять операторами, характерными для машинного решения.

Такие операторы (операторы переноса, переадресации, формирования, остановов и т. д.) называются операторами управления.

Схема решения, дополненная операторами управления, называется логической схемой программы.

Очевидно, что уровень описания схем решения является в формальном отношении высоким. Для этого используются алгоритмические языки, например АЛГОЛ-60.

Для описания логических схем программ применяются более конкретные и менее формальные, чем алгоритмические языки, уровни записи (промежуточные языки).

В настоящее время получили распространение две формы записи логических схем программ, или два метода программирования: блок-схемное программирование и операторный метод программирования.

Всем программистам известно:

ЭЦВМ любого типа предназначаются для реализации алгоритмических процессов;

конструкция ЭЦВМ любого типа рассчитывается на выполнение определенного набора элементарных операций.

Каждая операция машины имеет наименование и условный цифровой код, который называется кодом операции.

Операции в машине выполняются по командам.

Команда представляет собой совокупность кода операции и адресов машинных слов, которые участвуют в данной операции.

Команды различаются по своей структуре и состоят из трех функциональных частей:

адресной, предназначенной для записи (размещения) адресов машинных слов, результата операции и следующей команды;

операционной, содержащей код операции, которая выполняется по данной команде;

служебной, предназначенной для записи различных признаков, которые определяют порядок и способ использования остальных частей команды.

По количеству адресов, используемых в адресной части, команды могут быть одноадресными или многоадресными.

По типу выполняемых операций различаются команды:

- арифметических операций;
- операций преобразования кодов;
- операций управления;
- операций обмена информацией.

Надо иметь в виду, что все сведения для выполнения вычислений хранятся в запоминающих устройствах (ЗУ) ЭЦВМ. Эти устройства состоят из набора ячеек, которые занумерованы натуральными числами в восьмеричной системе счисления, называемыми номерами или адресами ячеек. Содержимое любой ячейки называется машинным словом и может быть числом или командой.

При выполнении некоторой команды управляющие сигналы «указывают» машине, какую операцию (код операции) и над какими машинными словами надо выполнить. При этом «указываются» не сами машинные слова, а адреса ячеек ЗУ, в которые они записаны.

Запись в адресных частях команд адресов ячеек, а не содержимого их обусловлена следующим:

при такой записи команды становятся универсальными, т. е. «обслуживают» различные исходные данные;

при этом можно оперировать неизвестными программисту результатами;

такая зависимость требует сравнительно небольшого количества разрядов для адресных частей команд.

Таким образом, командой называется специальный код, обеспечивающий выполнение ЭЦВМ одной из операций ее набора.

Определенная последовательность команд и чисел составляет программу решения данной задачи.

Программой называется последовательность команд, обеспечивающая решение данной задачи на конкретной ЭЦВМ.

Как правило, такая программа не только обеспечивает реализацию алгоритма решения задачи на конкретной ЭЦВМ, но и описывает более общий алгоритм, который содержит операции исходного алгоритма и некоторые вспомогательные операции формирования команд для выполнения заданного алгоритма.

Формирование машиной новых команд основано на реализации идеи «хранимой программы»: команды хранятся в одном массиве ячеек ЗУ с числами, имеют такой же вид, и машина над командами может выполнять те же действия, что и над числами.

Следовательно, любая команда программы в принципе доступна для анализа и переработки и в процессе решения задачи ее можно переадресовывать и формировать с помощью вспомогательных операций. Это обеспечивает задание программы в компактной форме, быструю выборку любой команды — высокое быстродействие ЭЦВМ.

Рассмотренное выше определение программы является частным случаем. В общем случае программой называется полное и точное описание на некотором формальном языке процесса переработки информации, приводящего к решению поставленной задачи.

Таким образом, программа служит источником информации об алгоритме и может быть использована для решения задачи на некоторой машине при наличии комплекса специальных программ, которые обеспечивают правильную интерпретацию информации исходной программы.

2.2. ОСНОВНЫЕ ПРИНЦИПЫ АЛГОРИТМИЗАЦИИ ЗАДАЧ

2.2.1. Алгоритм и его основные свойства

Как уже отмечалось, основой работы инженера-программиста является построение алгоритма процесса решения задачи, т. е. алгоритма обработки информации.

Любая задача решается выполнением определенной последовательности операций на ЭЦВМ, которая имеет конструктивный набор элементарных операций. Процесс обработки любой информации на ЭЦВМ реализуется по определенным правилам — алгоритмам.

Алгоритмом называется точное общепонятное предписание, которое определяет процесс преобразования исходных данных в искомый результат.

Алгоритм дает точное указание о выполнении в определенном порядке некоторой системы операций для решения задач данного класса. Это понятие сложилось в математике под влиянием трудов узбекского математика и астронома аль-Хорезми.

Примерами простейших алгоритмов являются правила выполнения четырех арифметических действий.

Для современной математики понятие алгоритма является одним из основных, так как теория алгоритмов, как ветвь математической логики, исследует возможность построения алгоритмов для решения любых задач. Поэтому алгоритмы должны представлять собой точные математические объекты.

Рассмотрим основные свойства алгоритмов.

Формальность применяемых правил означает применение их механически к исходным данным, без анализа действий, т. е. без выяснения вопроса, почему нужно делать так, а не иначе.

Например, для извлечения квадратного корня из натурального числа надо последовательно выполнить следующие действия (алгоритм извлечения квадратного корня):

а) разбить число на грани по две цифры, считая справа;

б) найти максимальный целый корень из первой грани слева, что и будет первой цифрой результата;

в) вычесть из этой грани квадрат первой цифры и к остатку снести вторую грань; полученное число условно называется первым остатком;

г) удвоить первую цифру результата и к тому, что получилось, приписать справа такую цифру, чтобы произведение образованного таким образом числа на эту цифру было бы максимальным целым числом, уместящимся в первом остатке; найденная цифра будет второй цифрой искомого результата;

д) вычесть полученное произведение из первого остатка и к полученной разности снести третью грань; образовавшееся число называется вторым остатком;

е) удвоить число, образованное двумя цифрами результата, и к тому, что получилось, приписать справа такую цифру, чтобы произведение образованного таким образом числа на эту цифру было максимальным целым числом, уместающимся во втором остатке. Найденная цифра будет третьей цифрой искомого результата и т. д.

Пример 1.

$$\sqrt{5\ 90\ 49} = 243$$

	4	
×	44	190
	4	176
×	483	14 49
	3	14 49

Из примера можно сделать вывод, что при извлечении квадратного корня не надо выяснять, почему операции выполняются именно в такой последовательности, и обосновывать правильность выполненных действий. В то же время полученный результат реализации алгоритма при возведении в квадрат доказывает, что корень найден правильно.

Следствием свойства формальности алгоритма является детерминированность, которая означает принятую последовательность решения. Следовательно, решение любой задачи по данному алгоритму может выполнить любой вычислитель (человек или устройство), если он умеет делать простейшие операции алгоритма. Результат вычислений не зависит от особенностей вычислителя, ему не обязательно понимать физический смысл решаемой задачи.

Результативность (эффективность) предполагает наличие таких вариантов исходных данных, для которых после конечного числа операций выполнение данного алгоритма заканчивается и дает искомый результат, и отсутствие таких вариантов исходных данных, для которых выполнение данного алгоритма приводит к неправильному результату.

Массовость выражает возможность применения данного алгоритма к различным вариантам исходных

данных, т. е. возможность решения любой задачи данного класса при наличии алгоритма. Например, алгоритм для нахождения двух чисел по их сумме и произведению заключается в следующем: необходимо составить квадратное уравнение приведенного вида так, чтобы коэффициент при неизвестном первой степени был равен сумме двух чисел с противоположным знаком, а свободный член — их произведению, и решить полученное уравнение. Найденные корни и будут искомыми числами.

Пример 2. Сумма двух чисел равна 6, произведение их равно 8. Составим квадратное уравнение: $x^2 - 6x + 8 = 0$ и решим его:

$$x_{1,2} = \frac{6}{2} \pm \sqrt{\frac{36}{4} - 8} = \frac{6}{2} \pm \frac{2}{2}; \quad x_1 = 4; \quad x_2 = 2.$$

Из примера видно, что данный алгоритм можно применять не только к указанным исходным данным, а к любым известным сумме и произведению двух чисел. Это свойство и есть массовость алгоритма.

К основным свойствам алгоритма нужно отнести еще два существенных для машинного счета:

— дискретность определяемого алгоритмом процесса, которая означает расчлененность алгоритма на отдельные последовательные операции (акты);

— простоту операций, выполняемых в каждом акте.

2.2.2. Алгоритмический процесс

Каждый алгоритм решения задачи применяется к определенным объектам — исходным данным, содержание которых определяется характером решаемой задачи. Это применение алгоритма называется алгоритмическим процессом, сущностью которого является переработка исходных данных по правилам данного алгоритма. Алгоритмический процесс состоит из ряда автономных актов переработки информации. При этом каждый акт переводит информацию из одного состояния в другое. Естественно, что результат применения каждого акта алгоритмического процесса к некоторой информации является исходной информацией для другого акта, который принимает управление (преемник). Некоторые акты алгоритмического процесса могут иметь несколько преемников (один и больше), поэтому задают-

ся правила однозначного определения преемника в зависимости от результата выполнения логического условия. Исходные данные, к которым применяется данный алгоритм, являются информацией для первого акта алгоритмического процесса.

Как уже отмечалось, программа решения задачи является описанием алгоритма решения с помощью системы команд конкретной ЭЦВМ, на «языке» машин.

Следовательно, каждая команда программы «описывает» акт переработки информации (вычислений) с определенным содержанием и его преемника. Программа описывает алгоритм, каждый акт которого содержит только одну элементарную операцию машины.

Таким образом, назначение ЭЦВМ состоит в применении алгоритма решения к исходным данным некоторой задачи, т. е. в реализации алгоритмического процесса.

Можно сделать следующие выводы:

- процесс алгоритмизации заключается в выделении актов обработки информации, записи в какой-то форме содержания каждого акта и определении порядка выполнения этих актов;

- вопрос о детальности описания алгоритма решается постепенно, начиная с «крупноблочной» структуры алгоритма; на начальных стадиях процесса алгоритмизации каждый акт обработки информации не доводится до одной команды (элементарная операция машины), а для изображения алгоритма применяются способы с хорошей обозримостью, например блок-схемы;

- процесс составления программы решения задачи можно разделить на две части:

- а) составление логической схемы программы;

- б) составление программы на «языке» конкретной машины; первая часть наиболее ответственна и является чисто творческой, трудно поддается формализации; здесь «язык» машины не применяется из-за специфической формы записи, трудного понимания и плохой обозримости; при удачном выборе формы записи логической схемы программы работа на второй части программирования может быть автоматизирована с помощью ЭЦВМ;

- сущностью программирования является алгоритмизация процесса переработки информации.

2.2.3. Основные законы (принципы) логики

Итак, разработка логической схемы процесса переработки информации — главная цель алгоритмизации. При построении алгоритма необходимо не только исходить из постановки и математического описания данной задачи, но и опираться на основные законы логики.

Основными законами логики считаются следующие принципы Аристотеля: закон тождества; закон противоречия; закон исключенного третьего и принцип Лейбница — закон достаточного основания.

Вот краткая характеристика этих законов.

Закон тождества

Определение этого закона в формальной логике: всякое A тождественно самому себе, или A есть A .

Такая формулировка является метафизическим пониманием этого закона. Но любое понятие отражает в сознании людей не абстрактные категории, а реально существующую материальную действительность.

В непрерывно развивающемся материальном мире даже в одном и том же предмете происходят непрерывные изменения:

«Истинное, конкретное тождество содержит в себе различие, изменение» (Ф. Энгельс «Диалектика природы»). Всякое тождество только в известном смысле является тождеством и вместе содержит элементы нетождества, т. е. различия.

«Диалектическая логика не отбрасывает просто тождество, она придает ему относительный смысл, приводит его в движение» [43].

Например, высказывание «артиллерийский полк есть артиллерийский полк» не дает никаких новых знаний о воинской части «артиллерийский полк». Это высказывание не определяет полностью обсуждаемый предмет: это артиллерийский полк периода первой мировой войны или это современный артиллерийский полк.

А какая характеристика артиллерийского полка обсуждается? Может быть, его огневые возможности? Или артиллерийский полк является объектом для перевозки по железной дороге? Поэтому, применяя принцип тождества к исследованию предметов или явлений, необходимо сначала определять сущность, отношение или

связь, которые интересуют исследователя в данном тождестве.

Таким образом, принцип тождества заключается в том, что «А (предмет, вещь) в известном данном отношении тождественно самому себе» [43].

При разработке алгоритмов решения задач на ЭЦВМ это позволяет проводить аналогии и отождествлять в определенном отношении предметы и явления, которые в повседневной жизни считаются различными.

Например, при построении алгоритма расчета транспортных средств для перевозки войск по железной дороге и составления графика движения войск воинские части (мотострелковые, танковые, артиллерийские полки и тыловые части соединений) целесообразно отождествлять как объекты для перевозок. Очевидно, что методика расчета перевозок для всех этих частей одинакова. Разница лишь в количественных характеристиках, которые задаются в виде переменных исходных данных для решаемой задачи.

Алгоритм расчета перевозок, составленный для одной воинской части, используется столько раз, сколько воинских частей назначается для перевозки.

Кроме того, алгоритмы обработки информации в системах управления войсками имеют сложную логическую схему программы обработки. Это вызвано разнообразными вариантами исходных данных и промежуточных результатов, которые необходимо учитывать при составлении алгоритмов решения задач управления. Принцип тождества требует указания определенного порядка обработки информации для каждого из этих вариантов. При этом для всех аналогичных вариантов порядок обработки информации должен быть одинаковым.

Таким образом, принцип тождества должен обеспечивать построение правильной и логически завершенной схемы обработки информации при составлении алгоритма и программы решения задач управления.

Закон противоречия

«...Невозможно вместе принимать, что та же самая вещь существует и не существует» или «невозможно, чтобы противоречащие утверждения были вместе истинными» (Аристотель. Метафизика. М.—Л., 1934).

Для диалектического понимания сущности этого принципа проанализируем два высказывания:

— артиллерийский полк — войсковая часть, которая выполняет боевую задачу;

— артиллерийский полк — не есть войсковая часть, которая выполняет боевую задачу, а является совокупностью грузов для перевозки по железной дороге.

По формально-логическому толкованию принципа противоречия эти высказывания противоречивы и истинным может быть одно из них. В реальной действительности оба высказывания одновременно могут быть истинными. Этот принцип необходимо применять в диалектическом понимании: «...два противоречащих одно другому суждения могут быть одновременно истинными, если в них выражаются противоречия действительности, основывающиеся на различных отношениях» [43].

Такое понимание принципа противоречия необходимо при разработке общей методики составления алгоритма и программы решения задач управления. При построении самого алгоритма не должны иметь место противоречащие друг другу в одном и том же отношении суждения или выводы.

В практической работе часто смешиваются действительные противоположности с противоположными определениями одной и той же сущности.

Например, при анализе возможных действий войск наступательным действиям противопоставляются оборонительные, т. е. считается, что понятию «наступать» противоречит понятие «обороняться». В реальных боевых условиях наступающим войскам противостоит и противодействует обороняющийся противник. Понятия «наступать» и «обороняться» — не действительные логические противоположности, а противоположные определения одной сущности.

Оба эти понятия выражают действия войск и в этом смысле сущность их тождественна. Логическими противоположностями являются понятия «наступать» и «не наступать».

Если при решении задачи по характеру действий войск установлено, что они не наступают, это не означает, что войска обороняются. Поэтому для определения характера действий войск необходимо выполнить всю совокупность обработки информации, получить соответ-

ствующие количественные или логические показатели, по которым и сделать выводы.

Нельзя смешивать истинные логические противоречия с противоположными определениями одной сущности при подготовке задач к решению на ЭЦВМ, так как при разработке методики машинного решения задач неоднозначность результатов решения (логических выводов) исключается.

Вот почему принцип противоречия считается одним из начал алгоритмизации задач и дополняется принципом исключенного третьего.

Закон исключенного третьего

В логике этот принцип (закон) объединяют с принципом противоречия, которое дополняется, детализируется им.

Сформулируем формально-логическое понимание принципа исключительного третьего. По принципу противоречия два противоречащие друг другу суждения не могут быть одновременно истинными, поэтому или одно, или другое истинно: третьей возможности нет.

Для повседневной практики такое понимание принципа справедливо, но для объяснения более сложных явлений явно недостаточно. Диалектическая логика дает такую формулировку принципа исключенного третьего: «...из двух противоречащих друг другу в одном и том же отношении суждений... или одно, или другое является истинным, и они исключают в том же самом отношении возможность истинности третьего суждения» [43].

Такое понимание принципа определяет границы его и делает не только применимым, но необходимым для научных и практических целей.

При составлении алгоритмов и программ решения задач на ЭЦВМ этот принцип надо строго соблюдать. Например, при решении многих практических задач управления на ЭЦВМ обычно встречаются различные логические условия, отношения которых определены заранее.

Большинство современных ЭЦВМ обеспечивает определение альтернатив по принципу «третий лишний», т. е. определяется только одна из двух ветвей вычислений (обработки информации). Поэтому алгоритм состав-

ляется по принципу исключенного третьего последовательным выделением одного из всех возможных путей (ветвей) вычислений, хотя таких ветвей может быть много.

Закон достаточного основания

Для познания объективного мира трех законов (принципов) Аристотеля оказалось недостаточно. Это понимал уже Г. Лейбниц и предложил принцип достаточного основания, который дает теоретическую базу для выделения частичной связи из универсальной.

«...Реальный смысл принципа достаточного основания в том, что выводы должны с необходимостью быть обоснованы и доказаны» [43].

Естественно, что каждый из логических выводов в любом алгоритме обработки информации должен иметь определенный количественный показатель, который и является его достаточным основанием.

Таким образом, алгоритм и программа решения задачи на ЭЦВМ составляются с учетом основных законов (принципов) логики при соблюдении всего комплекса логических условий, которые были выявлены на этапах постановки и математического описания задачи.

2.2.4. Последовательность построения алгоритмов

Еще раз напомним, что построение рабочего алгоритма предполагает наличие общей методики, которая разработана для решения алгоритмизируемой задачи.

Поэтому при алгоритмизации, т. е. при построении рабочего алгоритма не решаются вопросы:

- выбора количественных критериев;
- математического описания (моделирования) данной задачи;
- разработки общей последовательности преобразования исходных данных в искомый результат и другие вопросы той системы управления войсками, в интересах которой будет решаться алгоритмизируемая задача.

Ясно, что качество разработки задачи (процесса обработки информации) определяется глубиной научного исследования именно этих вопросов: они определяют степень объективного соответствия результатов решения

характеристикам реальных процессов и явлений, дают возможность получения научно обоснованных выводов по рассчитываемым количественным показателям.

Вот почему в современных условиях правильная научная формулировка постановки задачи должна отвечать ее содержанию и целям дальнейшего использования.

При разработке алгоритмов исследования математических моделей систем управления войсками необходимо учитывать зависимость функционирования этих систем от большого количества взаимосвязанных факторов, которые сами зависят от целого ряда других величин и условий.

Специфичность вносимых абстракций, большая и взаимно пересекающаяся комбинаторика посылок и следствий могут привести к потере основного содержания моделируемого явления, если для построения алгоритма моделей систем управления используются только математические и формально-логические категории.

При этом форма отображения реальной действительности в модели может затмить содержание и сущность явления на этапе алгоритмизации. И нельзя будет говорить о достижении требуемой степени адекватности процесса — модели процессу — действительности. Поэтому при разработке моделей систем управления войсками необходимо создавать алгоритмы нескольких классов, отличающихся наборами «элементарных действий». Тогда в алгоритме каждого класса описание одного и того же алгоритмического процесса будет являться «точным предписанием».

В алгоритме нулевого класса, т. е. начальном алгоритме, должна найти отражение структурная схема будущей модели. Главное назначение этой схемы — отобразить основное содержание моделируемого процесса, совокупность крупных функциональных компонентов его с основными показателями развития и возможные результаты.

«Элементарными действиями» нулевого класса целесообразно считать обобщенные в разумных пределах понятия и категории, применяемые в данной области знаний.

В структурной схеме модели бессмысленно переходить на язык математических формул. Здесь главное внимание необходимо сосредоточить на основном содер-

жании и целях моделирования. Тогда в дальнейшем упростится математическое описание отдельных процессов управления с сохранением физического, технического или военного смысла.

В структурной схеме оперируют обобщенными преобразованиями информации, которые непосредственно математически сформулировать не удастся. Это является причиной разработки самостоятельных алгоритмов для отдельных составных частей структурной схемы. Следовательно, могут быть разработаны алгоритмы первого, второго и последующих классов.

Естественно, что повышение класса алгоритма модели сопровождается все большей детализацией управления, используемого набора «элементарных действий» и конкретизацией их содержательного смысла.

На каком же классе алгоритма надо остановиться и каким должно быть содержание допустимых «элементарных действий» для разработки рабочего алгоритма? Дать однозначный ответ на это нельзя, так как все зависит от уровня сложности моделируемого процесса (явления), глубины его исследования, подготовленности и опыта исполнителей и других факторов.

Вот некоторые рекомендации, сделанные на основании опыта работы в этой области.

1. Если последовательность моделирования изображена на уровне некоторого набора «элементарных действий», то перед началом разработки алгоритма более высокого класса исполнитель обязан детально проанализировать каждый этап преобразования информации в составленном алгоритме. При этом анализ каждого блока должен начинаться с уяснения сущности, материальной основы соответствующего ему реального процесса, т. е. следует установить содержание и вид информации, которая должна быть получена на выходах анализируемого блока как результат обработки информации, подаваемой на его вход.

Затем надо выявить основные факторы, от которых зависят характер и результаты обработки информации. Эти факторы могут зависеть от большого количества других величин и иметь определенную взаимосвязь между собой вследствие сложности процессов, которые происходят в моделируемых системах.

Поэтому необходимо установить характер изменения

выявленных факторов в зависимости от других величин и условий, используемых и учитываемых в процессе моделирования.

2. Далее нужно попытаться разработать или подобрать формульные схемы, которые отражали бы зависимость основных факторов от всех других величин и условий. Если такие зависимости удастся математически сформулировать, то затем надо подобрать математический аппарат. Он должен обеспечивать получение рассчитываемых в анализируемом блоке величин по информации, поступающей на его вход с учетом всех основных факторов, которые определяют результаты данного этапа моделирования.

3. Если для каждого блока алгоритма возможно проделать весь комплекс перечисленных анализов и математически сформулировать характер преобразования информации, то можно утверждать, что используемый набор «элементарных действий» является достаточным для построения рабочего алгоритма модели.

Поэтому класс алгоритма, составленного с использованием «элементарных действий» такого содержания, можно считать окончательным.

4. Если будет установлено, что преобразование информации всеми или частью блоков алгоритма не может быть математически сформулировано, то разработка данного алгоритма не закончена и необходима его дальнейшая детализация.

Рассмотрим основные этапы подготовки задачи к решению на ЭЦВМ на примере описания алгоритмов с помощью АЛГОЛ-60.

2.3. КРАТКАЯ ХАРАКТЕРИСТИКА ОСНОВНОГО АЛГОРИТМИЧЕСКОГО ЯЗЫКА (АЛГОЛ-60)

2.3.1. Общие сведения об алгоритмических языках

В современных условиях количество публикаций (информации) в различных областях знаний по странам мира быстро растет. Хранение, систематизация, перевод и «библиографическое знакомство» с литературой — трудно разрешимая задача даже в специальной области знаний. Обычный (национальный) человеческий язык

избыточен для обмена информацией между «разноязычными» специалистами. Поэтому большой проблемой современной науки является обмен информацией между специалистами, т. е. проблема языка.

При обмене информацией между человеком и ЭЦВМ, которые получили широкое применение в человеческом обществе, выбор языка имеет очень важное значение.

Основными ограничителями возможностей современных ЭЦВМ являются объем памяти и пропускная способность внешних устройств. Создание универсального языка для обмена информацией между человеком и машиной — задача вычислительной математики и вычислительной техники.

Универсальный язык должен иметь минимальный словарный запас, что возможно при выполнении следующих требований.

1. Язык должен быть **однозначным**, т. е. каждому слову — строго определенный смысл.

2. **Суперпозиция** (слитное написание одного слова за другим) двух слов в языке не должна образовывать нового слова.

3. Язык должен иметь строгий синтаксис, т. е. систему правил, которые определяют допустимое написание отдельных слов и их последовательностей.

4. Язык должен допускать возможность **расширения**, т. е. введение новых слов при расширении областей применения.

5. Язык должен быть **универсальным**, т. е. пригодным для связи с любой ЭЦВМ, которая имеет транслятор, и обеспечивать максимальную простоту перевода.

В этих требованиях нашла отражение основная тенденция при создании языков, которая направлена на упрощение формы записи алгоритмов через наглядность, общедоступность и независимость от особенностей ЭЦВМ.

Эта тенденция привела к разработке специальных (проблемно-ориентированных) алгоритмических языков, которых уже разработано много.

В качестве международного алгоритмического языка был принят в 1960 г. на конференции в Париже АЛГОЛ-60. В 1963 г. в Киеве состоялась конференция представителей ВЦ СССР и стран народной демократии, на которой АЛГОЛ-60 принят как основной алгоритмический язык.

По мере повышения требований к языку повышались и требования к средствам его описания. Сейчас для описания алгоритмических языков пользуются нотацией Бэкуса (металингвистическими формулами). При этом используются следующие специальные символы:

специальные скобки $\langle \quad \rangle$, в которые заключаются понятия описываемого языка (лингвистические понятия), т. е. переменные величины;

$:=$ этот знак имеет смысл: по определению есть;

$|$ этот знак соответствует разделительному союзу ИЛИ. Если два или более языковых понятия записываются друг за другом без разделительного знака, то они должны следовать друг за другом в том порядке, в каком записаны.

При описании языковых понятий сначала описываются простые понятия, а затем с помощью уже введенных описаний более сложные. В металингвистических формулах использован метод рекурсивного определения, т. е. для определения той или иной конструкции языка применяется сама определяемая конструкция.

2.3.2. Назначение и структура АЛГОЛ-60

АЛГОЛ-60 предназначен для записи алгоритмов решения математических и инженерно-технических задач. Записи на этом языке называются АЛГОЛ-программами и используются для двух целей:

для автоматического составления программы решения задачи на ЭЦВМ; это осуществляется автоматическим переводом АЛГОЛ-программы с помощью заранее разработанного ТРАНСЛЯТОРА на язык конкретной ЭЦВМ;

для обмена АЛГОЛ-программами (алгоритмами) между ВЦ и НИИ.

Неизбежны различные варианты языка, т. е. разные условия применения его.

АЛГОЛ-60 был задуман на трех уровнях:

как **эталонный язык** — язык, который содержит перечень всех символов и синтаксических правил, определяет два других уровня языка и является базой при разработке входных языков и трансляторов для конкретных ЭЦВМ;

как язык публикаций, предназначенный для описания алгоритмов в печатных изданиях, поэтому предусматривающий расширение алфавита (например, использование греческого алфавита и т. п.);

как язык конкретных представлений (эталонный язык) — для использования эталонного языка при создании входных языков и трансляторов с учетом особенностей конкретных ЭЦВМ.

Структура языка определяется его основными элементами (конструкциями) и свойствами (рис. 2.1).

Всякий язык имеет свой алфавит и характеризуется синтаксисом и семантикой [9].

Алфавит АЛГОЛ-60 составляют основные символы (неделимые знаки). Более сложные конструкции (синтаксические единицы) языка являются последовательностями основных символов. Эти конструкции образуются по определенным правилам, совокупность которых составляет синтаксис языка.

Синтаксисом называется формальное задание записей на алфавите данного языка. Любая синтаксическая единица языка должна иметь вполне определенный семантический (содержательный) смысл, т. е. быть семантически стандартизована для общего использования.

Семантикой называется содержательный смысл синтаксических единиц (конструкций) языка.

По замыслу авторов АЛГОЛ-60 предназначался для описания методов обработки арифметической информации, т. е. для описания алгоритмов численного анализа.

Поэтому основной концепцией, которая используется в языке для описания алгоритмов вычислений, является понятие арифметического выражения. Составными частями (словами) арифметических выражений являются числа, переменные и функции, объединяемые знаками соответствующих операций.

Из арифметических выражений по правилам арифметической композиции образуются самостоятельные единицы языка — явные формулы, называемые операторами присваивания (вычислительные операторы). Для указания различных направлений (хода) вычислительных процессов в АЛГОЛ-60 добавлены неарифметические (логические, именующие) выражения, условные операторы и операторы цикла. С помощью этих конструкций языка можно описывать, например, альтернативы или



Рис. 2.1. Структурная схема АЛГОЛ-60

циклические повторения вычислительных операторов. Последовательность различных операторов можно заключать в операторные скобки (**begin** и **end**) для образования **составного оператора**. Для обеспечения функционирования различных операторов при необходимой взаимосвязи их соответствующие операторы могут снабжаться **метками**. Операторы дополняются **описаниями**, которые сами по себе не являются предписаниями о вычислениях алгоритма. Описания информируют транслятор о существовании и некоторых свойствах объектов, применяемых в операторах.

Например, класс чисел, используемых в качестве значений переменной, или размерность массива чисел, или совокупность правил, определяющих некоторую функцию. Последовательность описаний и следующая за ней последовательность операторов заключаются между операторными скобками (**begin** и **end**) и составляют **блок**, законченную алгоритмическую конструкцию языка.

Таким образом, АЛГОЛ-программа является **БЛОКОМ** или **СОСТАВНЫМ ОПЕРАТОРОМ**, если последний не содержится внутри другого оператора и сам не использует других операторов, которых нет в его составе.

Интересной алгоритмической конструкцией АЛГОЛ-60 является **ПРОЦЕДУРА**, с помощью которой осуществляется описание повторяющихся участков алгоритмов решения различных задач. В АЛГОЛ-программах процедуры имеют тот же смысл, что и стандартные подпрограммы (СП) при программировании для ЭЦВМ. Естественно, процедуры делают язык открытым, т. е. допускают возможность использования других кодов, кроме алфавита АЛГОЛ-60.

Таким образом, структура АЛГОЛ-60 напоминает обычную математическую символику с пояснениями (формульно-словесный способ). Поэтому запись алгоритмов решения задач на этом языке наглядна и понятна многим специалистам. В отличие от формульно-словесного способа АЛГОЛ-60 использует строгие правила записи алгоритмов, что обеспечивает точное, без всяких двусмысленностей, описание разнообразных вычислительных процессов и формальный перевод его на язык конкретной ЭЦВМ.

Кроме того, АЛГОЛ-60 дает возможность отразить

динамику вычислительного процесса. Вычисления описываются на АЛГОЛ-60 как развивающиеся во времени. Последовательность операторов в любой АЛГОЛ-программе соответствует порядку выполнения всех вычислительных операций алгоритма решения данной задачи.

Основные достоинства АЛГОЛ-60:

АЛГОЛ-60 является символическим языком программирования, близким к обычной математической символике с рядом оригинальных свойств;

этот язык имеет достаточную свободу для описания разнообразных алгоритмов и разумно обоснованные ограничения;

записи (АЛГОЛ-программы) на этом языке удобны и хорошо отражают структуру программы решения данной задачи на ЭЦВМ.

Кроме этих достоинств, АЛГОЛ-60 является хорошим средством связи при обмене научной информацией.

Все эти причины обусловили довольно широкое внедрение АЛГОЛ-60 в практическую работу вычислительных центров многих стран и создание производных алгоритмических языков (алголоидов).

2.3.3. Понятие об АЛГОЛ-программе и реализации ее на ЭЦВМ

АЛГОЛ-программой называется законченная и оформленная синтаксически запись алгоритма решения задачи на АЛГОЛ-60. Чаще всего АЛГОЛ-программы оформляются в виде блока. При этом программа состоит из последовательности описаний и операторов — в операторных скобках **begin** (начало) и **end** (конец). Все описания указываются в начале программы, а за ними — операторы. Описания и операторы разделяются между собой символом; (точкой с запятой). Главную алгоритмическую часть программы составляют операторы: только они содержат в себе указания о выполнении определенных этапов данного алгоритма. В АЛГОЛ-60 используются восемь видов операторов. Каждый из них предназначен для записи определенных структурных частей алгоритмов (рис. 2.2). Все операторы этого языка можно разделить на две группы:

ОСНОВНЫЕ ОПЕРАТОРЫ (присваивания, пустой, перехода, процедуры);

СОСТАВНЫЕ КОНСТРУКЦИИ (ЕДИНИЦЫ ДЕЙСТВИЯ)

Операторы <оператор>::= <безусловный оператор>|<условный оператор>|<оператор цикла>

I Безусловные операторы <безусловный оператор>::= <основной оператор>|<составной оператор>|<блок>

① **Основные операторы** <основной оператор>::= <непомеченный основной оператор>|<метка>;<основной оператор>

<непомеченный основной оператор>::= <оператор присваивания>|<оператор перехода>|<пустой оператор>|<оператор процедуры>

△ **Операторы присваивания**

<левая часть>::= <переменная>:=|<идентификатор процедуры>::=

<список левой части>::= <левая часть>|<список левой части><левая часть>

<оператор присваивания>::= <список левой части><арифметическое выражение>|<список левой части><логическое выражение>

△ **Операторы перехода**

<оператор перехода>::= go to <именующее выражение>

△ **Пустые операторы**

<пустой оператор>::= <пусто>

△ **Операторы процедур**

<оператор процедуры>::= <идентификатор процедуры><совокупность фактических параметров>

② **Составные операторы** <составной оператор>::= <непомеченный составной>|<метка>;<составной оператор>
<непомеченный составной>::= begin <конец составного>
<конец составного>::= <оператор> end |<оператор>;
<конец составного>

③ **Блоки** <блок>::= <непомеченный блок>|<метка>;<блок>
<непомеченный блок>::= <начало блока>;<конец составного>
<начало блока>::= begin <описание>|<начало блока>;<конец составного>

II Условные операторы

<оператор „если“>::= <условие><безусловный оператор>
<условный оператор>::= <оператор „если“>|<оператор „если“> else
<оператор>|<условие><оператор цикла>|<метка>;<условный оператор>

III Операторы цикла

<элемент списка цикла>::= <арифметическое выражение>|<арифметическое выражение> step <арифметическое выражение> until
<арифметическое выражение>|<арифметическое выражение> while <логическое выражение>

<список цикла>::= <элемент списка цикла>|<список цикла>,<элемент списка цикла>

<заголовок цикла>::= for <переменная>:= <список цикла> do
<оператор цикла>::= <заголовок цикла><оператор>|<метка>;<оператор цикла>

<ПРОГРАММА>::= <БЛОК>|<СОСТАВНОЙ ОПЕРАТОР>

Рис. 2.2. Блок-структура составных конструкций АЛГОЛ-60

II Описания *<описание> ::= <описание типа> | <описание массива> | <описание переключателя> | <описание процедуры>*

① Описание типа

<список типа> ::= <простая переменная> | <простая переменная>, <список типа>
<тип> ::= real | integer | Boolean
<локальный или собственный тип> ::= <тип> | own <тип>
<описание типа> ::= <локальный или собственный тип> <список типа>

② Описание массивов

<нижняя граница> ::= <арифметическое выражение>
<верхняя граница> ::= <арифметическое выражение>
<границная пара> ::= <нижняя граница> : <верхняя граница>
<список граничных пар> ::= <границная пара> | <список граничных пар>, <границная пара>
<сегмент массива> ::= <идентификатор массива> [<список граничных пар>] | <идентификатор массива>, <сегмент массива>
<список массивов> ::= <сегмент массива> | <список массивов>, <сегмент массива>
<описание массива> ::= array <список массивов> | <локальный или собственный тип> array <список массивов>

③ Описание переключателей

<переключательный список> ::= <именующее выражение> | <переключательный список>, <именующее выражение>
<описание переключателя> ::= switch <идентификатор переключателя> : = <переключательный список>

④ Описание процедур

<формальный параметр> ::= <идентификатор>
<список формальных параметров> ::= <формальный параметр> | <список формальных параметров>, <ограничитель параметра> <формальный параметр>
<совокупность формальных параметров> ::= <пусто> | (<список формальных параметров>)
<список идентификаторов> ::= <идентификатор> | <список идентификаторов>, <идентификатор>
<список значений> ::= value <список идентификаторов>; | <пусто>
<спецификация> ::= string | <тип> | array | <тип> array | label | switch | procedure | <тип>; procedure
<совокупность спецификаций> ::= <пусто> | <спецификация> <список идентификаторов>; | <совокупность спецификаций> <спецификация> <список идентификаторов>;
<заголовок процедуры> ::= <идентификатор процедуры> <совокупность формальных параметров>; <список значений> <совокупность спецификаций>
<тело процедуры> ::= <оператор> | <код>
<описание процедуры> ::= procedure <заголовок процедуры> <тело процедуры> | <тип> procedure <заголовок процедуры> <тело процедуры>

Рис. 2.3. Блок-структура описаний АЛГОЛ-60

ПРОИЗВОДНЫЕ ОПЕРАТОРЫ (составной, блок, условный, цикла).

Все основные операторы, а также производные — составной и блок — называются также **БЕЗУСЛОВНЫМИ ОПЕРАТОРАМИ**. Операторы АЛГОЛ-программы выполняются в естественном порядке, т. е. по записанной последовательности. При необходимости этот порядок выполнения операторов может изменяться с помощью оператора перехода.

Описания не содержат указаний о выполнении того или иного этапа алгоритма, а несут информацию о существовании и некоторых свойствах величин, примененных в операторах данной программы. Например, указывают класс чисел, размерность массивов и т. д. В АЛГОЛ-60 используются следующие виды описаний (рис. 2.3):

описание **ТИПА**;

описание **МАССИВОВ**;

описание **ПЕРЕКЛЮЧАТЕЛЯ**;

описание **ПРОЦЕДУРЫ**.

Операторы и описания, как основные синтаксические единицы языка, строятся из более простых его единиц — первичных конструкций, которые называются **ВЫРАЖЕНИЯМИ** (рис. 2.4).

В АЛГОЛ-60 используются следующие виды выражений:

АРИФМЕТИЧЕСКОЕ выражение;

ЛОГИЧЕСКОЕ (булевское) выражение;

ИМЕНУЮЩЕЕ выражение.

Выражения с помощью операций: арифметических, логических, отношения и следования — составляются из логических значений и **ЭЛЕМЕНТАРНЫХ КОНСТРУКЦИЙ** (рис. 2.5). К этим конструкциям относятся:

ИДЕНТИФИКАТОРЫ, ЧИСЛА, СТРОКИ, ПЕРЕМЕННЫЕ, УКАЗАТЕЛИ ФУНКЦИЙ.

Элементарные конструкции образуются непосредственно из **ОСНОВНЫХ СИМВОЛОВ** (рис. 2.6). Все указанные конструкции АЛГОЛ-60, их семантика и синтаксис подробно рассмотрены в сообщении о языке [9], а на рис. 2.2, 2.3, 2.4, 2.5, 2.6 с помощью металингвистических формул описан синтаксис этих конструкций.

Рассмотрим комплекс мероприятий, которые выполняются при решении задачи на ЭЦВМ с использованием

III. ПЕРВИЧНЫЕ КОНСТРУКЦИИ

а I Выражения <выражение> ::= <арифметическое выражение> | <логическое выражение> | <именующее выражение>

1 Арифметические выражения

<знак операции типа сложения> ::= + | -
<знак операции типа умножения> ::= x | ÷ | /
<первичное выражение> ::= <число без знака> | <переменная> | <указатель функции> | <арифметическое выражение>
<множитель> ::= <первичное выражение> | <множитель> * <первичное выражение>
<терм> ::= <множитель> | <терм> <знак операции типа умножения>
<простое арифметическое выражение> ::= <терм> | <знак операции типа сложения> <терм> | <простое арифметическое выражение> <знак операции типа сложения> <терм>
<условие> ::= if <логическое выражение> then
<арифметическое выражение> ::= <простое арифметическое выражение> | <условие> <простое арифметическое выражение> else <арифметическое выражение>

2 Логические выражения

<отношение> ::= <простое арифметическое выражение> <знак операции отношения> <простое арифметическое выражение>
<первичное логическое выражение> ::= <логическое значение> | <переменная> | <указатель функции> | <отношение> | <логическое выражение>
<вторичное логическое выражение> ::= <первичное логическое выражение> | <первичное логическое выражение>
<логический одночлен> ::= <вторичное логическое выражение> | <логический одночлен> ∧ <вторичное логическое выражение>
<логический терм> ::= <логический одночлен> | <логический терм> ∨ <логический одночлен>
<импликация> ::= <логический терм> | <импликация> ⇒ <логический терм>
<простое логическое выражение> ::= <импликация> <простое логическое выражение> ⇒ <импликация>
<логическое выражение> ::= <простое логическое выражение> | <условие> <простое логическое выражение> else <логическое выражение>

3 Именующие выражения

<метка> ::= <идентификатор> | <целое без знака>
<идентификатор переключателя> ::= <идентификатор>
<указатель переключателя> ::= <идентификатор переключателя> [<индексное выражение>]
<простое именующее выражение> ::= <метка> | <указатель переключателя> | <именующее выражение>
<именующее выражение> ::= <простое именующее выражение> | <условие> <простое именующее выражение> else <именующее выражение>

Рис. 2.4. Блок-структура выражений АЛГОЛ-60

II. ЭЛЕМЕНТАРНЫЕ КОНСТРУКЦИИ

① Идентификаторы

<идентификатор> ::= <буква> | <идентификатор> <буква> |
<идентификатор> <цифра>

② Числа

<целое без знака> ::= <цифра> | <целое без знака> <цифра>
<целое> ::= <целое без знака> | + <целое без знака> | - <целое
без знака>

<правильная дробь> ::= <целое без знака>

<порядок> ::= ₁₀ <целое>

<десятичное число> ::= <целое без знака> | <правильная
дробь> | <целое без знака> <правильная дробь>

<число без знака> ::= <десятичное число> | <порядок> | <деся-
тичное число> <порядок>

<число> ::= <число без знака> | + <число без знака> | - <число
без знака>

③ Строки

<чистая строка> ::= <любая последовательность основных
символов, не содержащая символа ' или символа "> | <пусто>

<открытая строка> ::= <чистая строка> | ' <открытая строка> '
<открытая строка> <открытая строка>

<строка> ::= ' <открытая строка>

④ Переменные

<идентификатор переменной> ::= <идентификатор>

<простая переменная> ::= <идентификатор переменной>

<индексное выражение> ::= <арифметическое выражение>

<список индексов> ::= <индексное выражение> | <список индек-
сов>, <индексное выражение>

<идентификатор массива> ::= <идентификатор>

<переменная с индексами> ::= <идентификатор массива> [<список
индексов>]

<переменная> ::= <простая переменная> | <переменная с индексами>

⑤ Указатели функций

<идентификатор процедуры> ::= <идентификатор>

<фактический параметр> ::= <строка> | <выражение> | <иден-
тификатор массива> | <идентификатор переключателя> |

<идентификатор процедуры>

<строка буквы> ::= <буква> | <строка буквы> <буква>

<ограничитель параметра> ::= , |) <строка буквы> :

<список фактических параметров> ::= <фактический пара-
метр> | <список фактических параметров> <ограничитель
параметра> <фактический параметр>

<совокупность фактических параметров> ::= <пусто> | (<спи-
сок фактических параметров>)

<указатель функции> ::= <идентификатор процедуры>

<совокупность фактических параметров>

Рис. 2.5. Блок-структура элементарных конструкций АЛГОЛ-60

1. ОСНОВНЫЕ СИМВОЛЫ ЭТАЛОННОГО ЯЗЫКА

<основной символ>::= <буква>|<цифра>|<логическое значение>|
<ограничитель>

① Буквы

<буква>::= a|b|c|d|e|f|g|h|i|j|k|l|m|
A|B|C|D|E|F|G|H|I|J|K|L|M|
n|o|p|q|r|s|t|u|v|w|x|y|z|

② Цифры

<цифра>::= 0|1|2|3|4|5|6|7|8|9

③ Логические значения

<логическое значение>::= true|false

④ Ограничители

<ограничитель>::= <знак операции>|<разделитель>|
<описатель>|<скобка>|<спецификатор>

⚠ Знаки операций

<знак операции>::= <знак арифметической операции>|
<знак операции отношения>|<знак логической
операции>|<знак операции следования>
<знак арифметической операции>::= +|-|*|/|÷|↑
<знак операции отношения>::= <|=|≤|≥|>|≠
<знак логической операции>::= ≡|=|∨|∧|¬
<знак операции следования>::= go to|if|then|else|for|do

⚠ Разделители

<разделитель>::= ,|.|;|:|{|}=|_|| step|until|while|comment

⚠ Скобки

<скобка>::= (|)|[|]|'|>| begin|end

⚠ Описатели

<описатель>::= own|Boolean|integer|real|array|switch|
procedure

⚠ Спецификаторы

<спецификатор>::= string|label|value

Рис. 2.6. Блок-структура основных символов АЛГОЛ-60

АЛГОЛ-60. Весь процесс решения задачи можно разделить на три этапа.

Первый этап. Алгоритм решаемой задачи описывается на АЛГОЛ-60. Составленная исходная АЛГОЛ-программа (ИАП) на специальных перфораторах наносится на перфоноситель (ПЛ, ПК). Исходные данные в состав ИАП не входят. Если в трансляторе предусмотрено, то ИАП с перфоносителя записывается на МЛ.

Второй этап. Транслятор переводит (транслирует) ИАП в рабочую программу на языке машины. Для этого в МОЗУ машины сначала вводится транслятор. За-

тем ему передается управление, он вводит ИАП в МОЗУ машины и перерабатывает в рабочую программу. Составленная на языке машины рабочая программа может быть оставлена в МОЗУ машины, выдана на перфоноситель или записана на МЛ. На этом этапе исходные данные в машину не вводятся, никаких вычислений не производится, т. е. рабочая программа еще не выполняется.

Третий этап. Выполняются вычисления по рабочей программе на ЭЦВМ. Для этого подготавливаются на перфоносителе исходные данные для ввода в ЭЦВМ. Управление передается рабочей программе, которая заранее отлаживалась и «прогонялась» на ЭЦВМ. Рабочая программа обеспечивает ввод исходных данных, процесс вычислений (обработки информации) и вывод результатов решения.

2.3.4. Элементы программирования на основном алгоритмическом языке

Об операторах ввода и вывода

Авторы основного алгоритмического языка (АЛГОЛ-60) не включили в эталонный язык специальных конструкций ввода и вывода информации. Это можно объяснить тем, что в современном мире выпускаются различные классы и типы ЭЦВМ с многоликими по характеру и возможностям устройствами ввода и вывода информации. Кроме того, для создателей трансляторов оставляется возможность самостоятельности в конкретном представлении языка (АЛГОЛ-60) при вводе и выводе информации. Авторы АЛГОЛ-60 дали рекомендации по использованию операторов ВВОДА и ВЫВОДА в ЭЦВМ. Запишем эти операторы с помощью металингвистических формул:

```
<оператор ввода> : := input (<список элементов>);  
<оператор вывода> : := output (<список элементов>);  
<список элементов> : := <элемент> | <список элементов>, <элемент>;  
<элемент> : := <переменная> | <идентификатор массива>.
```

Оператор ввода информации должен обеспечивать считывание информации с перфоносителя, перевод ее из

десятичной системы счисления в двоичную и присвоение ее значений соответствующим переменным (запись в ячейки).

Оператор вывода информации переводит информацию из двоичной системы счисления в десятичную и печатает значения переменных.

Программирование линейных вычислительных процессов и составление простейших АЛГОЛ-программ

На АЛГОЛ-60 алгоритмы линейных вычислительных процессов записываются последовательностью операторов присваивания. Главной частью оператора присваивания является выражение. Каждое выражение задает правило для вычисления некоторого значения. Классификация выражений связана с типом значений, определяемых с помощью данного выражения:

- значением арифметического выражения является число;
- значением логического выражения является логическое значение;
- значением именуемого выражения может быть только метка.

Каждое выражение может быть по своему содержанию (характеру) простым или условным.

Рассмотрим необходимые конструкции для программирования линейных вычислительных процессов.

Арифметическое выражение

Эта конструкция АЛГОЛ-60 задает правило для вычисления одного числового значения. Алгебраическое выражение — математический аналог арифметического выражения. Простое арифметическое выражение конструируется из чисел, переменных (простых и с индексами) и указателей функций с помощью знаков арифметических операций ($+$, $-$, $\times$, $/$, $\div$, $\uparrow$) и круглых скобок.

Правила записи арифметических выражений в основном аналогичны правилам записи алгебраических выражений. Особенность в том, что выражение АЛГОЛ-60 записывается в одну строку с использованием специаль-

ных знаков операций: возведения в степень $\uparrow$; деления / и деления нацело $\div$.

При вычислении значения одного арифметического выражения операции выполняются в последовательности слева направо с учетом старшинства: а) $\uparrow$; б) $\times$; /; $\div$; в) $+$; $-$.

Порядок выполнения операций можно изменять использованием круглых скобок. Значения выражений в круглых скобках вычисляются самостоятельно по указанному правилу и используются в дальнейших вычислениях. Значения указателей функций определяются перед выполнением с ними соответствующих операций.

Пример. Запишем в виде арифметического выражения АЛГОЛ-60 выражение:

$$\sqrt[3]{\frac{|a| - |b|}{2}} + \frac{3}{7} x_i^3 y_{jk}^2.$$

Решение:

$$((abs(a) - abs(b)) / 2) \uparrow (1 / 3) + \\ + 3 / 7 \times x[i] \uparrow 3 \times y[j, k] \uparrow 2$$

Тип значения выражения определяется типом значений составляющих его операндов.

Оператор присваивания

Очень часто этот оператор используется при программировании на основном алгоритмическом языке для записи обычных математических формул.

Действие этого оператора сводится к присваиванию некоторым величинам (переменным или идентификаторам процедур) значений, которые получены при вычислении арифметических или логических выражений.

Например, при выполнении оператора $y := b + 3$ к значению, которое в данный момент имеет переменная b , прибавляется 3 и результат присваивается переменной y . Прежнее значение переменной y теряется.

Из металингвистических формул определения оператора присваивания (рис. 2.2) видно, что значение выражения можно присваивать одновременно нескольким переменным:

$$\underbrace{y := z := a[k]}_{\text{список левой части}} := \underbrace{x \uparrow 3 + 2 \times \sin(x / 2)}_{\text{арифметическое выражение}}$$

Данный оператор присваивания соответствует формуле

$$y = z = a_k = x^3 + 2 \sin \frac{x}{2}.$$

При этом все переменные списка левой части должны иметь единый тип: только **integer** или только **real**, если правая часть — арифметическое выражение, или **boolean** при логическом выражении.

Если элементы списка левой части имеют одинаковый тип, но отличаются от типа значения арифметического выражения, то перед присваиванием необходимо автоматически выполнить преобразование:

а) если значение арифметического выражения E получает тип **real**, а переменные списка левой части тип **integer**, то преобразование сводится к округлению значения E до ближайшего целого по формуле **entier** $(E+0,5)$;

б) преобразование целых значений к типу **real** производится только при разном представлении в ЭЦВМ целых и вещественных чисел.

Оператор присваивания может не только присваивать начальное значение переменной, но и изменять ее значение. Например, оператор присваивания $n := n + 2$ увеличит значение n на две единицы. В этом наглядное различие между символом присваивания и знаком равенства.

Процесс присваивания значений с помощью оператора присваивания обычно выполняется в три этапа:

если в списке левой части переменные с индексами, то у них слева направо вычисляются значения всех индексных выражений;

вычисляется и преобразуется к типу левой части значение выражения из правой части;

полученное значение выражения присваивается всем переменным списка левой части, а индексы у переменных имеют значения, вычисленные на первом этапе.

Если в выражении правой части оператора присваивания какая-то переменная не имеет значения, то действие оператора не определено.

Описание типа

Все алгольные величины, с которыми производят действия в алгоритмах вычислительных процессов, обладают следующими свойствами:

- обозначаются с помощью идентификаторов;
- принимают значения строго определенного типа;
- имеют в АЛГОЛ-программе фиксированную область действия.

Информация об этих и других свойствах величин задается в описаниях программы. Каждый класс величин описывается определенным, связанным с ним видом описания. Например, характеристика простых переменных дается в описаниях типа.

Описание типа устанавливает, что в области его действия:

- перечисленные идентификаторы обозначают только простые переменные, а не какие-либо другие величины;
- указанные простые переменные принимают значения только того типа, который определен описанием.

Синтаксис описания типа определен соответствующими (рис. 2.3) металингвистическими формулами.

Примеры описаний типа:

integer i ; real y, z , $f2$; Boolean a, b ;

Приведенные описания типа имеют следующий смысл:

- идентификатором i обозначена простая переменная, которая может принимать лишь целые (положительные и отрицательные) значения, включая нуль;

- идентификаторами y, z и $f2$ обозначены простые переменные, которые в заданной области принимают только вещественные (положительные и отрицательные) значения, включая нуль;

- идентификаторами a и b обозначены простые переменные, которые в области действия описания могут принимать только логические значения **true** и **false**.

В списке типа размещаются в произвольном порядке простые переменные только одного типа и разделяются между собой запятой. Описания типа никаких конкретных значений переменным не присваивают. Например, в АЛГОЛ-программе есть описание типа **real π** (где π обозначена константа π). Это не означает, что переменная π имеет какое-то числовое значение. Присущее

сий числовое значение с требуемой точностью должно присваиваться в программе оператором присваивания. Например, $pi := 3.1415926$.

Описание массивов

В них задается информация о массивах переменных с индексами. Описание массива устанавливает следующее:

перечисленные в описании идентификаторы обозначают массивы, а не какие-то другие величины;

все элементы массивов принимают значения только одного типа, указанного в описании;

размерности массивов;

границы изменения индексов.

Диапазон значений, которые может принимать данный индекс у переменной, задается в описании массива граничной парой (рис. 2.3). Значения выражений, составляющих граничную пару, должны быть только типа *integer*. Например, для одномерного массива из двадцати элементов, пронумерованных от 1 до 20, граничная пара имеет вид 1:20. Нижняя граница не должна превышать верхней. Элементы массива всегда нумеруются последовательно от нижней границы с шагом +1 до верхней границы включительно. Индексы могут быть отрицательными. Например, граничная пара одномерного массива —4:3 означает, что восемь элементов массива пронумерованы: —4, —3, —2, —1, 0, 1, 2, 3. Границы изменения индексов многомерных массивов задаются в описании массива списком граничных пар, которые в списке разделяются запятой.

В описании массивов граничные пары и их списки записываются в индексных скобках.

Например: $[1:m]$; $[0:6, 0:4]$; $[1:n, 0:k, m:l]$.

Описание массива *integer array r [1:6]* определяет одномерный массив переменных $r_1, r_2, r_3, r_4, r_5, r_6$, элементы которого принимать могут только целые значения.

Для описания вещественных массивов можно описать тип *real* в описании массива не указывать. Например, описание массива *array a [1:m, 1:n]* определяет матрицу $\|a_{ij}\|$ вещественных чисел, имеющую m строк и n столбцов.

В АЛГОЛ-60 с помощью одного описания можно оп-

ределить несколько массивов. Если у различных массивов совпадают размерности, индексация элементов и тип значений для их элементов, то такие массивы при описании можно объединять в сегмент массива (рис. 2.3). Тогда в описании массивов указывается общий список граничных пар, а идентификаторы массивов разделяются запятыми. Например, описание массива `аггау` $a, b, c [1:7]$ определяет сразу три одномерных массива: a, b и c , каждый из семи элементов которых принимает вещественные значения.

Можно в одном описании объединять массивы только с одинаковым типом значений для их элементов. При этом сегменты описаний таких массивов объединяются в список массивов. Например, в описании массива `integer array` $a, b [1:m], i [1:l]$ список массивов состоит из двух сегментов:

- в первом определены два массива: a и b ;
- во втором — один массив i . Элементы всех трех массивов принимают целые значения.

Если элементы массивов принимают значения различных типов, массивы должны быть описаны в разных описаниях.

Заметим, что порядок размещения идентификаторов в сегменте, сегментов в списке массивов и нескольких описаний массивов между собой — произвольный. В описании массива, как и в описании типа, перед описателем типа может указываться описатель `own` (собственный).

Составление простых программ

Простейшие программы на АЛГОЛ-60 составляются в такой последовательности.

Выбирается необходимый порядок вычислений на основе анализа алгоритма решения задачи. Если есть повторяющиеся части формул, их обозначают отдельными идентификаторами и записывают в виде самостоятельных формул для повышения эффективности программы.

Для всех величин программы назначают идентификаторы. Последовательность формул алгоритма должна соответствовать записи последовательности операторов.

Устанавливается способ и порядок задания исходных данных и вывода результатов вычислений. При этом к моменту выполнения каждого оператора всем перемен-

ным, встречающимся в его правой части, необходимо присвоить исходные числовые значения. Присваивание исходных значений выполняется в АЛГОЛ-программе двумя способами:

— константам программы с помощью оператора присваивания;

— переменным с помощью оператора ввода.

Для выдачи результатов решения из машины в нужном месте программы записывается оператор вывода.

Выявляются идентификаторы для описания. В АЛГОЛ-программе должны описываться все идентификаторы, за исключением идентификаторов стандартных функций с общеизвестным смыслом, идентификаторов меток и идентификаторов формальных параметров процедур. Составляют описания и записывают их между собой в произвольном порядке, но обязательно перед последовательностью операторов.

Полученную последовательность описаний и операторов заключают в операторные скобки **begin** и **end**.

Пример. Составить АЛГОЛ-программу для вычисления значения полинома

$P_4(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ при любых значениях его коэффициентов и переменной x .

Значение полинома выдать на печать.

Решение. Обозначим значение полинома переменной y и преобразуем расчетную формулу по схеме Горнера:

$$y = (((a_4x + a_3)x + a_2)x + a_1)x + a_0.$$

АЛГОЛ-программа:

```
begin real x, y; array a [0 : 4];  
      input (x, a);  
y := (( (a[4] × x + a[3]) × x + a[2]) × x + a[1]) × x + a[0];  
      output (y)  
end;
```

ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

Для описания на основном алгоритмическом языке алгоритмов таких процессов используются следующие синтаксические единицы языка:

- логическое выражение;
- именуемое выражение;
- оператор перехода;
- оператор присваивания с условным арифметическим выражением в его правой части;
- условный оператор и другие конструкции.

Логическое выражение

Это выражение задает правило для вычисления одного логического значения: **true** или **false**. В АЛГОЛ-60 применяются простое и условное логические выражения (рис. 2.4).

Простое логическое выражение конструируется из логических значений, логических переменных, указателей логических функций и отношений с помощью знаков логических операций ($\neg$, $\wedge$, $\vee$, $\supset$, $\equiv$) и круглых скобок. Логические переменные и указатели логических функций принимают только логические значения. Часто используются простые логические выражения, состоящие только из отношений. Отношение — два простых арифметических выражения, соединенные знаком операции отношения.

Примеры отношений:

$$a=b; x \uparrow 2 + y \uparrow 2 \geq l \uparrow 2; \sin(x) < 0.$$

Значением отношения может быть одно из логических значений: **true** или **false**. Отношение принимает значение **true**, если после вычисления составляющих его арифметических выражений оно удовлетворяется. Иначе отношение принимает значение **false**. При вычислении значений простых логических выражений операции в пределах одного выражения выполняются последовательно слева направо с учетом порядка их старшинства: определяются значения отношений, $\neg$, $\wedge$, $\vee$, $\supset$, $\equiv$.

Последовательность выполнения операций можно изменить с помощью круглых скобок. Значения выражений (в круглых скобках) вычисляются самостоятельно по указанному правилу и используются в дальнейших вычислениях. Значения указателей логических функций определяются перед выполнением с ними соответствующих операций.

Пример. Записать логические выражения, которые истинны только при выполнении условий:

- а) переменная x принимает значения $0 \leq x < 1$;
- б) точка принадлежит первому квадранту плоскости $xу$.

Решение:

а) $x \geq 0 \wedge x < 1$; -

б) $x \geq 0 \wedge y \geq 0$.

Условное логическое выражение является конструкцией вида: $\langle \text{условие} \rangle \langle \text{простое логическое выражение} \rangle \text{ else } \langle \text{логическое выражение} \rangle$.

Металингвистическая формула условия: $\langle \text{условие} \rangle : := \text{if } \langle \text{логическое выражение} \rangle \text{ then}$. Часто логическое выражение условия представляет собой одно или несколько отношений, соединяемых знаками логических операций и выражающих физическую сущность условий в обычном смысле.

Примеры условий:

$$\text{if } x \neq 0 \text{ then; if } a + b < c \text{ then; if } y > 0 \wedge x > 0 \text{ then.}$$

Условие выполняется, если его логическое выражение принимает значение **true**, и не выполняется, если принимает значение **false**. Таким образом, условие определяет, что надо взять в качестве значения исходного условного логического выражения:

а) если условие выполняется, то значением условного логического выражения будет значение простого логического выражения, стоящего за условием;

б) иначе — значение логического выражения (за символом **else**).

Например, значением условного логического выражения

$$\text{if } x \geq y \text{ then } a \wedge b \text{ else } a \vee b$$

будет:

а) при $x \geq y = \text{true}$ значение простого логического выражения $a \wedge b$;

б) при $x \geq y = \text{false}$ значение логического выражения $a \vee b$. С помощью оператора присваивания значение любого логического выражения может быть присвоено одной или нескольким переменным. Например:

$$b := \text{true};$$

$$a1 := d := \neg (x < y) \wedge \text{true};$$

$$n := \text{if } x \neq y \text{ then } \neg a \text{ else } a \vee b;$$

Условное арифметическое выражение

При записи алгоритмов часто встречаются формулы, в которых значение некоторой величины зависит от выполнения различных условий и определяется разными формулами, например:

$$y = \begin{cases} x & \text{при } x \geq 0; \\ x^2 + 1 & \text{при } x < 0. \end{cases} \quad (1)$$

В АЛГОЛ-60 для описания таких формул используется условный оператор или оператор присваивания с условным арифметическим выражением. Конструкция условного арифметического выражения аналогична конструкции условного логического выражения (рис. 2.4).

Например: `if  $a \geq 0$  then  $x + y$  else  $x - y$ .`

Значение, которое после вычислений принимает условное арифметическое выражение, определяется условием. Если оно выполняется, то этим значением будет значение простого арифметического выражения (за условием). Если оно не выполняется, то этим значением будет значение арифметического выражения (за символом `else`). С помощью оператора присваивания значение условного арифметического выражения можно присвоить одной или нескольким переменным.

Например, формулу (1) можно записать:

$$y := \text{if } x \geq 0 \text{ then } x \text{ else } x \times x + 1.$$

В конструкцию условного арифметического выражения после символа `else` записывается любое арифметическое выражение, которое по синтаксису АЛГОЛ-60 может быть простым или условным. Следовательно, условные арифметические выражения могут как бы «вкладываться» друг в друга.

Например:

$$\underbrace{\text{if } B1 \text{ then } A1 \text{ else } \underbrace{\text{if } B2 \text{ then } A2 \text{ else } A}_{\text{условное арифметическое выражение}}}_{\text{условное арифметическое выражение}}$$

«Вложенные» условные арифметические выражения применяются для описания формул, в которых значение

некоторой величины определяется по трем и более различным выражениям.

Обращаем внимание на то, что в конструкции условного арифметического выражения за условием между символами **then** и **else** может записываться только простое арифметическое выражение. Но синтаксис АЛГОЛ-60 обеспечивает преобразование любого условного выражения в простое с помощью круглых скобок.

Приведем пример:

if B1 then (if B2 then A2 else A1) else A

простое арифметическое
выражение (получено
из условного)

Оператор перехода.
Именующее выражение

В АЛГОЛ-60 принят естественный порядок выполнения операторов, т. е. слева направо в порядке их написания. Но при описании алгоритмов очень часто возникает необходимость нарушения естественного порядка выполнения операторов программы. Для этого в основном алгоритмическом языке предназначен оператор перехода (рис. 2.2), который указывает, откуда и куда передается управление при выполнении АЛГОЛ-программы. Для выделения оператора-преемника управления из последовательности операторов используется конструкция языка — метка. Каждый оператор программы может выделяться одной или несколькими метками. Метками могут быть идентификаторы или целые числа без знака. Примеры меток: *m1*, *vv*, *st12*, 144.

Метка ставится перед оператором и отделяется от него двоеточием. Например, требуется меткой *M1* выделить оператор присваивания $i := i + 2$. Для этого необходимо записать: $M1:i := i + 2$. Правило для определения метки задается именующим выражением, т. е. значением последнего является метка. Применяются простые и условные именующие выражения (рис. 2.4).

Простое именующее выражение — отдельная метка или указатель переключателя. Условное именующее выражение по конструкции аналогично условному логическому выражению.

Например:

if $x < 0$ then Q else R1.

Значением условного именуемого выражения является метка, определяемая простым именуемым выражением (— за условием, которое выполняется), иначе — именуемым выражением (—за символом else).

Таким образом, после вычисления именуемого выражения оператор перехода приводится к виду `goto <метка>`.

Например, `goto 113; goto 152`.

Оператор перехода записывается на том месте АЛГОЛ-программы, где изменяется естественный порядок выполнения операторов. По метке, полученной в результате вычисления именуемого выражения оператора перехода, отыскивается помеченный ею оператор, который должен принять управление от данного оператора перехода и выполняться. Поэтому метки ставятся только перед теми операторами программы, которые получают управление (выполняются) при нарушении естественного порядка выполнения программы. Идентификаторы меток в АЛГОЛ-программе не описываются.

Условный оператор

Данный оператор относится к группе производных операторов, т. е. содержит в своем составе другие операторы и предназначен для организации их работы.

Операторы, которые входят в состав других операторов, принято называть подоператорами. Условный оператор (рис. 2.2) бывает трех видов:

сокращенный условный оператор (оператор «если»);
полный условный оператор;

сокращенный условный оператор (с подоператором цикла).

Действие оператора «если» в программе сводится к следующему.

При выполнении условия (логическое выражение принимает значение true) выполняется подоператор оператора «если» и затем или оператор-преемник подоператора, или оператор, следующий за оператором «если».

Например, `if  $x=0$  then goto M; S;`

При невыполнении условия оператор «если» не вы-

зывает никакого действия в программе, а будет выполняться следующий оператор.

Пример. Составить АЛГОЛ-программу для вычисления $y = n!$ с выдачей на печать значений n и y .

Решение:

```
begin integer n, y, i;  
  input (n);  
  i := 0; y := 1;  
M1: i := i + 1; y := y × i;  
  if i < n then goto M1;  
  output (n, y)  
end;
```

Полный условный оператор включает в свой состав не один, а два подоператора и вызывает действия в программе при выполнении и при невыполнении условия.

Например, **if $a = 0$ then goto M1 else goto N2; S;**

Действие этого оператора сводится к выполнению подоператора (безусловного оператора), стоящего за условием, или к выполнению подоператора (— за символом **else**). Затем выполняется оператор-преемник (если он определен) или оператор, стоящий за полным условным оператором.

В конструкцию полного условного оператора после символа **else** может записываться любой оператор, в том числе и условный. Следовательно, условные операторы могут как бы «вкладываться» друг в друга.

if B1 then S1 else if B2 then S2 else P,
⏟
условный оператор
⏟
условный оператор

где P вновь может быть условным оператором.

Действие такого сложного условного оператора сводится к выполнению подоператора, стоящего за первым слева выполненным условием, или к выполнению подоператора, который записан за последним символом **else**, если ни одно из условий не выполняется.

Действие в программе сокращенного условного оператора с подоператором цикла аналогично действию оператора «если».

Составной оператор

При описании алгоритмов обработки информации на АЛГОЛ-60 часто необходимо группу операторов синтаксически рассматривать как единый оператор. Для объединения группы операторов в единый оператор в АЛГОЛ-60 предусмотрен составной оператор (рис. 2.2), который получается с помощью операторных скобок **begin** и **end**. Между этими скобками записывается объединяемая группа операторов, дающая составной оператор.

Пример. Необходимо записать условным оператором алгоритм определения большего и меньшего из неравных друг другу чисел x и y . Обозначим: a — большее число; b — меньшее число.

Решение:

if $x > y$ then begin $a := x; b := y$ end
 else begin $a := y; b := x$ end;

Рассмотрим вычислительный процесс с достаточно сложной разветвляющейся схемой.

Пример. Составить АЛГОЛ-программу перевода прямоугольных координат x, y в полярные r, φ ($0 \leq \varphi < 2\pi$) по формулам:

$$r = \sqrt{x^2 + y^2};$$

$$\varphi = \begin{cases} 0 & \text{при } x=0; y=0; \\ \arctg \frac{y}{x} & \text{при } x > 0; y \geq 0; \\ \frac{\pi}{2} & \text{при } x=0; y > 0; \\ \pi + \arctg \frac{y}{x} & \text{при } x < 0; \\ \frac{3\pi}{2} & \text{при } x=0; y < 0; \\ 2\pi + \arctg \frac{y}{x} & \text{при } x > 0; y < 0. \end{cases}$$

Решение. Если значение $\left| \arctg \frac{y}{x} \right|$ близко к $\frac{\pi}{2}$, т. е. когда $y \gg x$, то при вычислении на ЭЦВМ воз-

возможен контрольный останов. Поэтому при $y > x$ надо $\arctg \frac{y}{x}$ вычислять через дополнительный угол $\arctg \frac{x}{y}$.

Введем идентификаторы:

pi — для числа π ; fi — для угла φ ; p — для угла $\arctg \frac{y}{x}$.

АЛГОЛ-программа:

```
begin real x, y, pi, r, fi, p; input (x, y);
pi := 3.1415926; r := SQRT (x × x + y × y);
if x = 0 then begin fi := if y = 0 then 0
else if y > 0 then pi/2 else 1.5 × pi;
goto m1 end;
if abs(x) ≥ abs(y) then p := arctan (y/x)
else begin p := arctan (x/y);
p := (pi/2 + abs (p)) × sign (p) end;
fi := if x < 0 then pi + p else if y ≥ 0
then p else 2 × pi + p;
m1 : output (r, fi)
end
```

Программирование циклических вычислительных процессов

Алгоритмы циклических вычислительных процессов на основном алгоритмическом языке можно описать с помощью операторов присваивания, перехода и условного оператора. Но записи получаются довольно сложными, громоздкими и мало наглядными. Для удобства и увеличения эффективности записи алгоритмов циклических вычислительных процессов в АЛГОЛ-60 введен специальный оператор цикла.

Структура оператора цикла

Металингвистическая формула оператора цикла (рис. 2.2):

$\langle \text{оператор цикла} \rangle := \langle \text{заголовок цикла} \rangle \langle \text{оператор} \rangle$.

Заголовком цикла называется конструкция:

$\text{for} \langle \text{переменная} \rangle := \langle \text{список цикла} \rangle \text{ do}$.

Переменную заголовка цикла иногда называют параметром цикла. Заголовок цикла управляет повторениями исполнения подоператора цикла, т. е. оператора, который входит в состав оператора цикла. Заголовок цикла устанавливает, что с каждым значением параметра цикла, предусмотренным списком цикла, необходимо исполнить подоператор цикла. Естественно, что подоператором цикла может быть любой оператор АЛГОЛ-60. Если в цикле необходимо повторно исполнять группу операторов, то их надо объединить в составной оператор.

Список цикла состоит из одного или нескольких элементов списка цикла, разделяемых запятыми. Каждый элемент списка цикла устанавливает значения, которые присваиваются переменной цикла. При этом элементы в списке цикла перебираются в порядке их написания. После того как список цикла будет исчерпан, в естественном порядке выполнится оператор, следующий за оператором цикла.

АЛГОЛ-60 использует три разновидности элементов списка цикла:

- арифметическое выражение;
- типа арифметической прогрессии;
- типа пересчета (итерационного типа).

Рассмотрим особенности выполнения операторов цикла с разновидными элементами списка цикла. Для упрощения записи введем обозначения:

- i — параметр (переменная) цикла;
- $A_1, A_2, \dots$ — арифметические выражения;
- L — логическое выражение;
- P — подоператор цикла.

Элемент типа арифметической прогрессии

Этот элемент применяется для описания алгоритмов циклических вычислительных процессов, в которых параметр цикла с начального значения принимает несколько последовательных значений с постоянным шагом изменения.

Оператор цикла с одним элементом списка цикла этого типа имеет вид

for $i := A_1$ step A_2 until A_3 do P .

Для переменной i , изменяющейся от $A1$ с шагом $A2$ до $A3$ включительно, выполнять оператор P .

Пример. Составить АЛГОЛ-программу вычисления и печати таблицы значений функции

$$y = \frac{x \operatorname{tg} \frac{x}{2}}{1 + x^2}$$

для значений x , изменяющихся от $x_0 = 1,5$ с шагом $\Delta x = 0,5$ до $x_n = 12,5$ включительно.

Решение:

```
begin real x, y;
  for x := 1.5 step. 5 until 12.5 do
    begin y := x * sin (x/2) / cos (x/2) / (1 + x ↑ 2);
      output (x, y) end
    end;
```

Элемент списка цикла итерационного типа

Этот элемент применяется для описания алгоритмов циклических вычислительных процессов итерационного характера, т. е. когда последующие значения переменной цикла получаются из предыдущих по некоторому закону.

Оператор цикла с одним элементом списка итерационного типа (типа пересчета) имеет вид

```
for i := A1 while L do P
```

«Для переменной i , изменяющейся по закону $i := A1$, пока истинно L , выполнять оператор P ».

Пример. Составить АЛГОЛ-программу алгоритма вычисления $y = \sqrt{x}$ по формуле Герона:

$$y_{i+1} = 0,5 \left(y_i + \frac{x}{y_i} \right),$$

если

$$y_0 = \frac{1+x}{2}$$

с точностью

$$|y_{i+1} - y_i| < \varepsilon.$$

Решение:

```
begin real  $x, eps, y_0, y_1$ ;  
  input ( $x, eps$ );  
   $y_0 := .5 \times (1 + x)$ ;  
for  $y_1 := (x/y_0 + y_0) \times .5$  while  $abs(y_1 - y_0) \geq eps$  do  
   $y_0 := y_1$ ;  
  output ( $x, y_0$ )  
end;
```

Примечания: 1. При использовании элемента этого типа начальное значение переменной цикла должно присваиваться перед оператором цикла.

2. Один элемент типа пересчета, как и элемент типа арифметической прогрессии, задает переменной цикла несколько значений.

Арифметическое выражение — элемент списка цикла

Элемент этого типа применяется при невозможности определения общего закона изменения переменной цикла. Поэтому в списке цикла обычно перечисляются все конкретные значения переменной цикла или закон получения каждого из них.

Оператор цикла с одним таким элементом списка цикла имеет вид

for $i := A1$ do P .

Его действие эквивалентно выполнению двух операторов: $i := A1$; P . Каждый элемент этого типа задает переменной цикла только одно значение и обеспечивает выполнение подоператора P всего один раз. Список цикла может состоять из одного или из нескольких элементов. При этом в списке могут объединяться элементы различных типов.

Элемент списка цикла «арифметическое выражение» обычно используется в комбинации с элементами того же или других типов.

Пример. Составить АЛГОЛ-программу для вычисления и печати значений функции

$$y = \frac{\sqrt[3]{\cos x}}{1 + \sin 0,5 x}$$

при следующих значениях аргумента $x = 1,53; 2,15, 2,79; 3,11$.

Решение:

```
begin real x, y;  
  for x: = 1.53, 2.15, 2.79, 3.11 do  
begin y: = cos(x) ↑ (1/3)/(1 + sin(x/2)); output (x, y)  
  end  
end;
```

Вложенные циклы

Часто встречаются циклические вычислительные процессы, в которых циклы могут в своем составе иметь один или несколько других циклов. Такие циклы называются вложенными. Для описания алгоритмов обработки информации с вложенными циклами используется оператор цикла, подоператор которого также является оператором цикла. В таких операторах с каждым фиксированным значением переменной внешнего цикла внутренний цикл выполняется при всех значениях его переменной.

Пример. Составить АЛГОЛ-программу вычисления и печати таблицы значений полинома $P(x) = a_8x^8 + a_7x^7 + \dots + a_1x + a_0$ при заданных $a_0, a_1, \dots, a_8$ и значениях x , меняющихся от $x_0 = 1$ с шагом $\Delta x = 0,025$ до $x_n = 2$ включительно.

Решение:

Для вычисления значений полинома применим схему Горнера.

АЛГОЛ-программа:

```
begin integer i; real x, p; array a [0 : 8];  
  input (a);  
  for x: = 1 step.025 until 2 do  
    begin p: = a[8];  
    for i: = 7 step -1 until 0 do  
      p: = p × x + a[i]; output (x, p)  
    end end;
```

Примечания: 1. Метка внутри оператора цикла недоступна оператору перехода, который расположен вне оператора цикла.

2. Выход из оператора цикла возможен до полной выборки списка цикла. Для этого внутри подоператора цикла должен быть оператор перехода, который выводит к метке вне оператора цикла.

Программа. Блок.

Запись алгоритмов процессов обработки информации завершается оформлением АЛГОЛ-программы. По синтаксису программа является блоком или составным оператором, который не содержится внутри другого оператора и не использует других операторов, не содержащихся в нем.

Поясним это примером. Рассмотрим АЛГОЛ-программу вычисления координат центра тяжести системы n материальных точек, расположенных на плоскости, с координатами x_i, y_i и массами m_i .

АЛГОЛ-программа:

```
A: begin integer n; input (n);
  B: begin real M; array m[1 : n]; integer i;
      input (m); M: = 0.0;
    for i: = 1 step 1 until n do M: = M + m[i];
    C: begin real X; array x[1 : n]; input (x);
      X: = 0.0;
    for i: = 1 step 1 until n do X: = X + m[i] × x[i];
      X: = X/M; output (X)
    end;
    D: begin real Y; array y[1 : n]; input (y);
      Y: = 0.0;
    for i: = 1 step 1 until n do Y: = Y + m[i] × y[i];
      Y: = Y/M; output (Y)
    end end end;
```

При составлении этой программы учитывались вопросы экономного использования МОЗУ машины, так как n может быть большим числом. Программа оформлена в виде блока, где A — внешний блок, B — его подблок. Подблоками блока B являются независимые друг от друга блоки C и D .

Значение описаний в АЛГОЛ-программе

Мы уже рассмотрели общее назначение описаний в АЛГОЛ-программе и дали характеристику описаниям типа и массивов.

Определим область действия описаний. Областью дей-

ствия описаний является блок, в начале которого размещены эти описания. Каждый блок АЛГОЛ-программы с помощью описаний определяет величины, которые существуют и действуют только внутри этого блока. При этом любая величина обозначается идентификатором, указанным в ее описании. Вне данного блока описанная в нем величина не существует, а идентификатор — ее обозначение может применяться для наименования любой другой величины. Следовательно, каждый идентификатор локализуется в данном блоке соответствующим описанием. Иначе: смысл идентификатор приобретает при входе в блок и теряет при выходе из него. Конечно, в начале блока каждый идентификатор описывается только один раз, т. е. в одном блоке различные величины нельзя обозначать одним идентификатором.

Некоторые идентификаторы и обозначенные ими величины являются глобальными по отношению к блоку, в котором они действуют, но не описаны. Естественно, что они должны быть описаны в одном из внешних блоков, в состав которого входит данный блок (подблок) — область действия глобальных величин.

Иногда одним и тем же идентификатором в программе обозначаются разные величины, одни из которых описываются во внешнем блоке, другие — во внутреннем. В этом случае величина, описанная во внешнем блоке, не может применяться во внутреннем блоке, так как при входе в него идентификатору этой величины придается другой смысл. Во внутреннем блоке данный идентификатор обозначает другую величину. После выхода из внутреннего блока этот идентификатор получает прежний смысл, который он имел во внешнем блоке.

Таким образом, описания связывают ту или иную величину с определенным идентификатором и определяют область ее действия в АЛГОЛ-программе. В соответствующих блоках обязательно должны быть описаны все идентификаторы, применяемые в данной АЛГОЛ-программе. Если величины обозначены неописанными идентификаторами, они считаются неопределенными и просто не существуют в данной программе.

В АЛГОЛ-60 разрешается не описывать идентификаторы:

— стандартных функций (они имеют постоянный, заранее установленный смысл во всех программах);

— формальных параметров процедур.

Блочная структура построения программ и принцип локализации идентификаторов при разработке АЛГОЛ-программ обеспечивают следующие преимущества:

а) возможность разработки программы по частям различными сотрудниками независимо друг от друга; при такой организации работы каждый сотрудник получает глобальные идентификаторы для своей части задачи; выбор идентификаторов для локальных величин в каждой части программы каждый сотрудник выполняет произвольно и независимо от других сотрудников, так как эти идентификаторы описываются в своих самостоятельных блоках; все это значительно облегчает организацию программирования больших и сложных задач;

б) возможность динамического распределения памяти в ходе решения задачи на ЭЦВМ. Это приводит к экономному использованию памяти и осуществляется следующим образом: определенное количество ячеек памяти ЭЦВМ для хранения значений закрепляется за величинами, которые указаны в описаниях блока на период его выполнения, т. е. после входа в блок через его начало и до выхода из него; после выхода из блока прекращают существование с потерей своих значений величины, описанные в нем; освободившееся поле памяти ЭЦВМ используется для размещения значений других величин.

Иногда необходимо сохранение полученных значений локальных величин при выходе из соответствующего блока. Для этого АЛГОЛ-60 имеет собственные величины. Описания таких величин снабжаются дополнительным описателем *own* («собственный» — перевод с английского). Основной символ *own* размещается в описании перед описателем типа величины.

Например:

```
own real a, b;  
own integer array m[1 : 9].
```

При повторном входе в соответствующий блок значения собственных величин остаются неизменными с момента первого выхода из этого блока. Значения же несобственных локализованных в этом блоке величин при первом выходе из него теряются и при повторном входе в блок не определены. Для использования этого свойства

собственных величин программа составляется так, чтобы при первом входе в блок ей присваивалось некоторое исходное значение, а при повторных входах это присваивание пропусклось. Это достигается обычно с помощью глобальной для данного блока величины.

Пример. Пусть исследуется точность вычислений функции

$$y = e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!} + \dots$$

в зависимости от числа i членов ряда.

Необходимо составить АЛГОЛ-программу с выдачей на печать величины члена ряда при каждом i и полученного значения функции.

Решение:

```

M1 : begin integer i, n; real x, y;
      input (x, n);
      y := 1;
      for i := 1 step 1 until n do
        M2 : begin own real u;
              if i = 1 then u := 1;
              u := u × x/i; y := y + u; output (i, u, y)
            end end;

```

В блоке M2 переменная u описана как собственная. При первом входе в блок M2 при $i=1$ ей будет присвоено значение 1. В дальнейшем при $i>1$ это присваивание будет обходиться и новые значения переменной u будут получаться из старых. Обход присваивания исходного значения при повторных входах в блок M2 достигается с помощью переменной i , описанной в блоке M1, и глобальной для блока M2.

Необходимо обратить внимание на важный практический случай, когда в описаниях массивов границы у индексов непостоянны. Тогда в состав граничных пар входят идентификаторы, а граничные пары являются арифметическими выражениями.

Например, array [1:n+2]; integer array r[1:m×n].

В этом случае значения арифметических выражений граничных пар надо вычислять еще при входе в соответствующий блок при обработке данного описания, что-

бы определить длину массива и распределить для него определенное количество ячеек памяти ЭЦВМ. Для этого идентификаторы граничных пар должны быть глобальными для данного блока (описаны во внешнем блоке) и им должны быть присвоены значения еще до входа в блок, в котором используются такие описания массивов.

Метки и примечания в АЛГОЛ-программе

АЛГОЛ-60 не устанавливает для меток специальных описаний. Но принцип локализации для описываемых в блоке величин распространяется и на метки. Факты использования метки являются как бы описанием ее. Метка, отделенная от оператора двоеточием, действует так, как будто она описана в начале младшего по алгоритмической иерархии блока, в котором расположен данный, помеченный оператор. Поэтому метка недоступна операторам перехода, которые расположены за пределами блока ее локализации.

Принцип локализации меток означает, что войти в блок можно только через его начало, так как внутренние метки недоступны извне.

В одном блоке не должно быть двух и более операторов с одинаковыми метками. А в разных блоках одинаковой меткой могут быть обозначены различные операторы. Выйти из блока можно не только естественным путем после выполнения последнего его оператора, но и с помощью оператора перехода к метке одного из операторов внешнего блока.

АЛГОЛ-60 разрешает включение в текст АЛГОЛ-программ примечаний для пояснения сущности алгоритма. Текст примечаний при трансляции программы не воспринимается и не влияет поэтому на ее выполнение. Применяются два способа включения примечаний в АЛГОЛ-программу. При первом способе используется специальный разделитель `comment`, а текст примечания записывается за этим разделителем. Разделитель `comment` разрешается записывать в программе после символа точка с запятой или после символа `begin`. После текста примечания обязательно ставится точка с запятой. Текст примечания может содержать любые символы любого

алфавита, кроме точки с запятой. При втором способе текст примечания записывается в программе за символом **end** АЛГОЛ-программы. При этом пояснительный текст (примечания) не должен иметь в своем составе точку с запятой, **end** и **else**. Поэтому текст примечаний можно записывать между символами: **end** и; **end** и **else**; **end** и **end**.

Пример. Записать на АЛГОЛ-60 алгоритм определения в прямоугольной матрице вещественных чисел $a_{ij}[m \times n]$ максимального из минимальных элементов каждой строки и его координат (номеров строки и столбца).

Решение:

```

begin integer  $m, n$ ;

    input ( $m, n$ );

    begin integer  $i, j, k, l, s$ ; real max, min;
    array  $a[1 : m, 1 : n]$ ; comment здесь обозначены:
    max — наибольший из  $\min_i$ , min — наименьший
        элемент в строке,
     $k$  — номер строки и  $l$  — номер столбца max;
        input ( $a$ ); max: = 0.0;
        for  $i$ : = 1 step 1 until  $m$  do
        begin min: =  $a[i, 1]$ ;  $s$ : = 1;
        for  $j$ : = 2 step 1 until  $n$  do
            if  $\min > a[i, j]$  then
                begin min: =  $a[i, j]$ ;  $s$ : =  $j$ 
        end цикла по  $j$  — поиск минимального элемента min
            в строке;
            if  $\max < \min$  then
                begin max: = min;  $k$ : =  $i$ ;  $l$ : =  $s$ 
                end
        end цикла по  $i$  — поиск элемента max;
        output (max,  $k, l$ )
        end
    end;
```

Процедуры

Часто алгоритмы различных задач имеют одинаковые участки для вычисления по одним и тем же формулам, но с разными исходными данными в том или ином месте процесса обработки информации. При составлении программ решения таких задач целесообразно операторы повторяющегося участка процесса обработки информации описать один раз, присвоить им общее наименование и затем использовать в требуемых местах программы. При этом каждый раз необходимо указывать только наименование этого участка и новые исходные данные для обработки информации.

При машинном программировании такие участки оформляют в виде подпрограмм.

В основном алгоритмическом языке понятию подпрограммы соответствует понятие процедуры. Под словом процедура понимается определенная совокупность действий по обработке информации, приводящая к искомому результату. Эта совокупность рассматривается как единое целое. Надо иметь в виду, что в алгоритмах обработки информации очень часто встречаются кроме процедур вычислительного характера процедуры по обработке информации без получения значений (процедуры ввода, вывода информации, преобразования матриц и т. д.).

АЛГОЛ-60 имеет специальные конструкции для задания процедуры и обращения к процедуре для ее выполнения. Задание процедуры осуществляется через описание процедуры. Обращение к процедуре может выполняться двумя способами:

- в виде оператора процедуры;
- в виде указателя функции.

Описание процедуры

Это описание служит для задания процедуры. Оно, как и любое другое описание (типа, массивов, переключателя), размещается в начале того блока, в котором выполняется обращение к данной процедуре. Описание процедуры само по себе является пассивным элементом АЛГОЛ-программы и не вызывает в ней никаких действий, хотя основную часть этого описания составляют операторы.

Процедуры, принятые в алгоритмическом языке в качестве стандартных (элементарные функции анализа, ввода, вывода информации и др.), употребляются в программах без описаний.

Металингвистическая формула описания процедуры:

$\langle \text{описание процедуры} \rangle ::= \text{procedure } \langle \text{заголовок процедуры} \rangle$

$\langle \text{тело процедуры} \rangle$

Если перед символом **procedure** в описании процедуры указывается наименование типа: **integer**, **real** или **Boolean**, то процедура описывается как процедура-функция.

Основной частью описания процедуры является тело процедуры, которое задает последовательность действий (алгоритм), выполняемых при обращении к данной процедуре.

Тело процедуры оформляется так:

$\langle \text{тело процедуры} \rangle ::= \langle \text{оператор} \rangle \mid \langle \text{код} \rangle$

Чаще всего тело процедуры записывается в виде блока. $\langle \text{Код} \rangle$ — не конструкция АЛГОЛ-60. Это произвольная запись, смысл которой в конкретном входном языке устанавливается заранее дополнительным соглашением.

Введение $\langle \text{кода} \rangle$ расширяет возможности АЛГОЛ-60. Некоторые операции и алгоритмы нельзя эффективно записать на АЛГОЛ-60, например операции над частями машинных слов, вычисления с удвоенной точностью и ряд других. Обычно такие процедуры записываются в $\langle \text{коде} \rangle$ машины, т. е. с помощью ее системы команд. При трансляции тело процедуры, записанное на языке машины, включается в общую рабочую программу без изменения в виде ее отдельной части.

Пусть в виде процедуры необходимо записать алгоритм определения суммы:

$$S = \sum_{i=k}^n a_i.$$

Для этого в тело процедуры надо включить два оператора:

$s := 0.0;$
for $i := k$ **step** 1 **until** n **do** $s := s + a[i].$

Но тело процедуры оформляется в виде одного оператора. Поэтому эти два оператора должны быть оформлены в составной или в блок.

Анализ назначения процедуры и операторов, реализующих ее алгоритм, приводит к выводу, что идентификаторы в теле процедуры по своей роли различны. Например идентификатор i обозначает порядковый номер слагаемого суммы, не зависит от конкретного вида вычисляемой суммы, имеет локальное назначение и поэтому может быть описан в теле процедуры. Идентификаторы k , n , a , s в теле процедуры выполняют формальную роль. Они не обозначают конкретные объекты суммы, а показывают те позиции в теле процедуры, которые при ее выполнении должны занимать соответствующие им по смыслу реальные величины. Такие идентификаторы тела процедуры называются формальными параметрами, которые в АЛГОЛ-60 не описываются.

Следовательно, тело процедуры суммирования должно быть оформлено в виде блока:

```
begin integer i; S: = 0.0;  
for i: = k step 1 until n do S: = S + a[i]  
end;
```

где k , n , a , s — формальные параметры;

i — идентификатор, локализованный в теле процедуры.

В заголовке процедуры указывается наименование (идентификатор) процедуры, перечисляются формальные параметры и приводится поясняющая их информация:

$\langle \text{заголовок процедуры} \rangle : := \langle \text{идентификатор процедуры} \rangle$

$\langle \text{совокупность формальных параметров} \rangle$;

$\langle \text{список значений} \rangle \langle \text{совокупность спецификаций} \rangle$

Из этой металингвистической формулы видно, что заголовок процедуры разделяется точкой с запятой на две части. Первая часть заголовка процедуры обязательно включается в описание каждой процедуры, а вторая — может отсутствовать.

Совокупность формальных параметров записывается в круглые скобки в виде списка формальных параметров, которые разделяются между собой запятыми. Для обозначения формальных параметров и названия про-

цедуры можно использовать произвольные идентификаторы. Приведем описание процедуры суммирования, тело которой уже записано в виде блока:

```
procedure sum (k, n, a, s);  
begin integer i; S:=0.0;  
for i:=k step 1 until n do S:=S+a[i]  
end;
```

Оператор процедуры

Обращение к процедуре выполняется с помощью оператора процедуры:

$\langle \text{оператор процедуры} \rangle := \langle \text{идентификатор процедуры} \rangle$

$\langle \text{совокупность фактических параметров} \rangle$.

Совокупность фактических параметров — записанная в круглые скобки последовательность фактических параметров, которые отделены друг от друга запятыми.

Фактическими параметрами могут быть:

- выражения;
- идентификаторы массивов;
- идентификаторы переключателей;
- идентификаторы процедур.

Количество и порядок следования фактических параметров в операторе процедуры должны строго соответствовать количеству и содержательному смыслу формальных параметров в описании соответствующей процедуры.

Пусть в АЛГОЛ-программе, в которой имеется описание процедуры (рассматривалось выше), требуется вычислить сумму:

$$y = \sum_{j=1}^m x_j.$$

Для этого в нужном месте АЛГОЛ-программы записывается оператор обращения к процедуре SUM:

... ; SUM (1, m, x, y); ...

Оператор процедуры будет выполняться в следующем порядке:

а) по идентификатору оператора процедуры SUM отыскивается соответствующее описание процедур; фак-

тические параметры ставятся в соответствии с формальным в порядке их следования;

б) в теле процедуры формальные параметры заменяются соответствующими фактическими; в результате такой замены параметров получается модифицированное тело процедуры;

в) модифицированное тело процедуры выполняется в АЛГОЛ-программе на месте оператора процедуры. Так, в приведенном примере после модификации тела процедуры будут выполняться операторы:

$y := 0.0;$
for $i := 1$ **step** 1 **until** m **do** $y := y + x[i];$

Оператор процедуры определен только в области действия описания процедуры. Поэтому должны быть определены, описаны в соответствующей области, величины, которые входят в состав фактических параметров.

Процедуры-функции

В результате выполнения некоторой процедуры может определяться только одно значение. Такую процедуру целесообразно оформлять в виде процедуры-функции. Описание процедуры-функции имеет следующие особенности:

— в заголовке процедуры перед символом **procedure** обязательно указывается тип (**integer**, **real** или **Boolean**); это — своего рода описание некоторой простой переменной, обозначенной идентификатором процедуры; поэтому последний не должен дополнительно описываться в теле процедуры или указываться среди формальных параметров;

— в теле процедуры должен быть хотя бы один оператор присваивания, в левой части которого есть идентификатор данной процедуры. Следовательно, вычисленное значение функции должно быть присвоено идентификатору процедуры.

Например, алгоритм вычисления суммы

$$S = \sum_{i=k}^n a_i$$

можно записать в виде процедуры-функции:

real procedure SUM (i, k, n, a);

```

begin real s; s: = 0.0;
for i: = k step 1 until n do s: = s + a[i];
SUM: = S
end;

```

Обращение к процедуре-функции выполняется только с помощью указателя функции:

$\langle \text{указатель функции} \rangle := \langle \text{идентификатор процедуры} \rangle$
 $\langle \text{совокупность фактических параметров} \rangle$.

Указатель функции является составным элементом арифметического или логического выражения. Например, в АЛГОЛ-программе, которая содержит описание процедуры-функции суммирования, необходимо вычислить

$$X = \frac{\sum_{i=1}^n m_i x_i}{\sum_{i=1}^n m_i}.$$

Это можно выполнить с помощью оператора присваивания:

$X := \text{SUM}(i, 1, n, m[i] \times x[i]) / \text{SUM}(i, 1, n, m[i]),$

в котором обращение к процедуре суммирования выполнено в виде указателей функции.

Оператор процедуры, как самостоятельная законченная единица языка, не может быть составной частью выражений и размещается в последовательности операторов.

Любую процедуру-функцию можно при необходимости представить в виде процедуры общего типа. Но когда определяются значения нескольких величин в результате выполнения некоторой процедуры, ее нельзя описывать в виде процедуры-функции.

Процедура без параметров

В теле процедуры можно применять идентификаторы трех видов:

- идентификаторы, обозначающие формальные параметры;
- идентификаторы, локализованные в теле процеду-

ры; при этом тело процедуры оформляется в виде блока;

в) идентификаторы, которые обозначают величины, не локализованные в теле процедуры — они описываются в блоке, охватывающем описание данной процедуры.

В описании процедуры могут отсутствовать формальные параметры. В этом случае в теле процедуры используются только идентификаторы первого и третьего вида (a и b). Такие процедуры называются процедурами без параметров. Их можно описывать и в общем виде, и в виде функций. Заголовок процедуры без параметров имеет вид:

<тип> procedure <идентификатор процедуры>
или **procedure <идентификатор процедуры>**.

При обращении к процедуре без параметров указывается только ее идентификатор. Например, при решении задачи исследуется движение материальной точки по некоторой траектории. При этом в различных местах программы решения задачи неоднократно определяется ее удаление от начала координат по формуле

$$d = \sqrt{x^2 + y^2 + z^2}.$$

При описании на АЛГОЛ-60 алгоритма решения этой задачи целесообразно вычисление расстояния оформить в виде процедуры-функции без параметров:

real procedure d;
d := sqrt (x×x+y×y+z×z);

В этом описании процедуры все идентификаторы x , y , z являются глобальными для тела процедуры и должны быть описаны в блоке программы, охватывающем данное описание процедуры. При обращении к данной процедуре в виде указателя функции d будет вычислено расстояние при тех значениях координат x , y и z , которые они имеют в момент обращения.

Характеристика параметров процедуры

Как уже отмечалось, заголовок процедуры разделяется точкой с запятой на две части. Что же указывается во второй части заголовка? Здесь описывается информация, поясняющая некоторые свойства формальных па-

раметров процедуры. Она необходима для облегчения работы транслятора и повышения качества получаемой при трансляции рабочей программы. Например, в теле процедуры встречается выражение $x \uparrow y$, в котором x и y являются формальными параметрами. Какие команды для вычисления этого выражения надо включить в рабочую программу при трансляции? Если y — переменная типа `real`, то в рабочей программе реализуется формула

$$x \uparrow y = e^{y \ln x},$$

если y — типа `integer`, то x надо умножить само на себя y раз.

Если в описании процедуры нет информации о формальных параметрах, то транслятор для определения типа переменной должен предварительно проанализировать все обращения к данной процедуре в АЛГОЛ-программе. Если информация о типе формального параметра указана, то обращения к процедуре анализировать не надо и трансляция значительно упрощается. Вторая часть заголовка процедуры включает:

<список значений> <совокупность спецификаций>.

Совокупность спецификаций (рис. 2.3) с помощью описателей `real`, `integer`, `Boolean`, `array`, `switch`, `procedure` и спецификатора `label` задаст информацию о классах и типах формальных параметров.

С помощью одной спецификации можно пояснить одновременно несколько формальных параметров, которые при этом объединяются в список идентификаторов. Например: `real x, y, z; integer array a, b; label p11; procedure range.`

Спецификации по своей роли в АЛГОЛ-программе и по способу записи имеют много общего с описаниями. Все формальные параметры рекомендуется специфицировать для упрощения работы транслятора. Если в процедуре какому-то формальному параметру могут соответствовать фактические параметры разных классов, то подобные формальные параметры нельзя включать в совокупность спецификаций.

Обязательно специфицируются те формальные параметры, которые входят в список значений заголовка процедуры. Надо помнить, что в совокупности специфика-

ций каждый формальный параметр может использоваться только один раз.

Список значений заголовка процедуры предназначен для перечисления идентификаторов тех формальных параметров, которым при модификации тела процедуры передаются значения соответствующих им фактических параметров.

При модификации тела процедуры при замене формальных параметров наименованиями (выражениями) соответствующих фактических параметров иногда получаются малоэффективные рабочие программы и даже синтаксически недопустимые конструкции. Например, в АЛГОЛ-программе описан алгоритм Евклида в виде процедуры. Сущность алгоритма нахождения общего наибольшего делителя двух целых, не равных нулю чисел: число x делится на число y и, если остаток $r \neq 0$, то делитель становится делимым, а остаток — делителем. Процесс деления повторяется до получения $r = 0$, при этом последний делитель и есть искомый результат.

Запишем эту процедуру:

```
procedure evklid ( $x, y, r$ );  
  integer  $x, y, r$ ;  
  m1 : if  $y = 0$  then  $r := x$  else  
begin  $r := x - x \div y \times y$ ;  $x := y$ ;  $y := r$ ; goto m1  
end;
```

Если обратиться к этой процедуре с помощью оператора evklid (37, $a+b$, d), то после модификации тела процедуры должен быть выполнен оператор:

```
  m1 : if  $(a + b) = 0$  then  $d := 37$  else  
begin  $d := 37 - 37 \div (a + b) \times (a + b)$ ;  $37 := (a + b)$ ;  
       $(a + b) := d$ ; goto m1  
end;
```

в подоператоре которого получены недопустимые с точки зрения синтаксиса конструкции: $37 := (a + b)$ и $(a + b) := d$.

Поэтому в АЛГОЛ-60 предусмотрен еще один способ модификации тела процедуры, когда в тело передаются не сами фактические параметры, а только их значения. Для этого и предназначен в заголовке процедуры список значений:

$\langle \text{список значений} \rangle := \text{value} \langle \text{список идентификаторов} \rangle \mid \langle \text{пусто} \rangle$.

Идентификаторы формальных параметров, перечисленные в списке значений, сохраняются в модифицированном теле процедуры. Присваивание им значений фактических параметров производится непосредственно перед входом в тело процедуры. При этом тело процедуры как бы охватывается дополнительным блоком, в начале которого и выполняется это присваивание.

Естественно, в список значений можно включать только те формальные параметры, у которых соответствующие им фактические параметры могут принимать значения. Из всех возможных классов фактических параметров такими являются выражения и идентификаторы массивов. Идентификаторы процедур и переключателей, как не принимающие значений, передаются в тело процедуры только через замену наименований.

Формальные параметры, включенные в список значений, обязательно специфицируются. Например, в процедуре определения общего наибольшего делителя целесообразно в список значений включить формальные параметры x и y :

```
procedure evklid ( $x, y, r$ );  
value  $x, y$ ; integer  $x, y, r$ ;  
m1 : if  $y = 0$  then  $r := x$  else  
begin  $r := x - x \div y \times y$ ;  $x := y$ ;  $y := r$ ; goto m1  
end;
```

При обращении к этой процедуре через оператор evklid (37, $a+b$, d) модифицированное тело процедуры примет вид:

```
begin integer  $x, y$ ;  
 $x := 37$ ;  $y := a + b$ ;  
m1 : if  $y = 0$  then  $d := x$  else  
begin  $d := x - x \div y \times y$ ;  $x := y$ ;  $y := d$ ; goto m1  
end  
end;
```

Процедуры алгоритмических языков широко применяются при описании типичных процедур обработки военной информации (см. главы 3 и 4).

Глава 3

ОСНОВНЫЕ МЕТОДЫ И ПРИЕМЫ ОБРАБОТКИ БОЛЬШИХ МАССИВОВ ВОЕННОЙ ИНФОРМАЦИИ НА ЭЦВМ

В современных условиях совершенствование планирования и управления боевыми действиями войск в вооруженных силах всех стран зависит от уровня развития технических средств и организации обработки военной информации.

В последние годы ЭЦВМ, СКМ, СПМ и другие технические средства управления во многих типовых вариантах поступают на снабжение в органы управления (штабы) армий многих стран.

Накоплен большой опыт практического применения ЭЦВМ в управлении войсками противовоздушной обороны, противоракетной обороны, полевой артиллерии и других организационных структур армии США.

Естественное развитие получила комплексная обработка используемой информации: от исходных данных до выдачи документов в органы управления (штабы). Зарубежные специалисты считают, что только комплексный, системный, т. е. кибернетический, подход к сбору, обработке и использованию военной информации может обеспечить эффективное управление войсками. Эта идея нашла свое наиболее яркое воплощение в автоматизированных системах управления войсками (АСУВ). Конечно никакая АСУВ не может полностью заменить оперативное, боевое и политическое руководство войсками. Решая различные классы задач управления, АСУВ призвана обеспечивать соответствующие органы управления (штабы) систематизированной информацией, которая необ-

ходима для ускорения и принятия обоснованных решений.

Таким образом, автоматизированные системы в общем процессе управления войсками должны выполнять важную управленческую функцию, функцию обработки информации.

Ядром, основой любой АСУВ являются информационные массивы. Только они объединяют в единое информационное целое все звенья и решаемые задачи системы. От структуры и организации информационных массивов зависит оперативность и достоверность результатов, выдаваемых системой, т. е. эффективность управления.

Оптимальное формирование, рациональная обработка и комплексное использование информационных массивов — основная системно-техническая задача организации любой системы управления войсками и проектирования АСУВ.

3.1. КЛАССИФИКАЦИЯ ОСНОВНЫХ МЕТОДОВ ОБРАБОТКИ БОЛЬШИХ МАССИВОВ ИНФОРМАЦИИ НА ЭЦВМ

3.1.1. Вычислительные и логические процессы обработки информации

Основное применение ЭЦВМ в системах управления войсками — решение комплексных задач управления; при этом предполагается автоматизация обработки больших информационных массивов.

Процессы обработки больших массивов военной информации по характеру применяемых методов разделяются на вычислительные и логические (рис. 3.1).

Вычислительные процессы применяются для обработки числовой информации. По смысловому содержанию обрабатываемой информации методы обработки имеют следующую классификацию:

- интерполирование;
- численное дифференцирование;
- вычисление интегралов;
- решение систем линейных алгебраических уравнений;



Рис. 3.1. Классификационная схема методов решения комплексных задач

- решение нелинейных уравнений;
- аппроксимация функций;
- численное интегрирование обыкновенных дифференциальных уравнений;
- численное решение уравнений в частных производных;
- статистическое моделирование;
- исследование моделей боевых действий.

Логические процессы применяются для обработки нечисловой информации, которая представляется комбинациями букв и цифр в определенном коде. Отметим характерные особенности логической обработки информационных массивов:

- использование больших объемов информации, что требует рационального распределения различных уровней памяти и емкости ее;
- нечисловой характер исходных данных и результатов, что требует развитой системы устройств ввода — вывода и разработки специальных программ выдачи результатов.

Задачи логической обработки информационных массивов принято называть информационно-логическими задачами (ИЛЗ). Эти задачи решаются в целях упорядочения в массиве информации и выдачи справок по различным запросам на основе этой информации. В качестве примера ИЛЗ можно привести задачу по сбору сведений о материально-техническом снабжении воинских частей различными средствами. Для решения этой задачи необходимо хранить в памяти машины и систематически обновлять сведения о поступлении в части материальных средств и их расходе, выдавать справки об обеспеченности и потребностях частей, формировать исходные данные для расчета заявок на снабжение.

Информационные массивы для логической обработки на ЭЦВМ обычно состоят из отдельных записей, которые называют еще формулярами или сообщениями.

Запись — это определенный набор данных, характеризующих некоторый объект или процесс. Каждая запись состоит из отдельных символов или группы символов с конкретным смысловым содержанием. Такие структурные части записи называются элементами. Ими могут быть числа, буквы, комбинации букв и цифр, спе-

циальные символы для обозначения размерности (например, градусы, проценты и т. п.).

Примеры записей:

— строка в бланке укомплектованности личным составом воинских частей или весь бланк;

— сводка об обеспеченности воинской части различными видами довольствия и т. п.

Естественно, что однородные записи со сведениями об однородных объектах целесообразно объединять в массивы: информационные поля или таблицы.

Таблица 4

№ воинской части	Укомплектованность офицерским составом, %	Укомплектованность рядовым и сержантским составом, %
$\begin{matrix} a \\ b \end{matrix}$	$\begin{matrix} m \\ n \end{matrix}$	$\begin{matrix} p \\ q \end{matrix}$

Рассмотрим простейший вид информационного массива — информационную таблицу характеристик (ИТХ). В этой таблице (табл. 4) содержатся сведения об укомплектованности личным составом воинских частей. Каждая строка ИТХ является записью. Первый элемент этой записи — наименование объекта, сведения о котором содержатся в записи, а остальные элементы — значения различных характеристик для соответствующего объекта. Поэтому при размещении подобных таблиц в памяти ЭЦВМ отводится место (поле памяти) для записей, составляющих таблицу. Описание таблицы: наименования характеристик; способ представления значений характеристик; структура записей (количество символов в каждом элементе записей), — можно записывать в память ЭЦВМ или же учесть при программировании задач, в которых обрабатывается данная таблица.

При обработке на ЭЦВМ информационных таблиц основными проблемами являются:

— изменение значений характеристик в отдельных записях по новым сообщениям о состоянии объектов информации;

— выдача справок о части информации из таблицы

или обобщенных сведений по одной (нескольким) характеристике данной таблицы;

— расширение (сокращение) данных таблицы и построение новых ИТХ.

При логической обработке на ЭЦВМ информационных массивов важное значение имеют следующие проблемы:

— отыскание в информационных массивах записей с заданными значениями отдельных характеристик;

— рациональное размещение в памяти машины массивов сложной структуры с учётом нескольких уровней запоминающих устройств с различной ёмкостью и временем выборки. При этом элементы записей обычно не совпадают по количеству разрядов с ячейками памяти ЭЦВМ.

Для практического решения этих проблем целесообразно выбирать рациональные методы организации хранения, сортировки, поиска информации и сочетание этих методов.

В практике работы зарубежных и отечественных вычислительных центров накоплен богатый арсенал методов логической обработки информационных массивов [18]. Настало время классифицировать их и разработать методику рационального применения.

Авторы в своей практике проанализировали многие методы и выделили их в следующие классы (рис. 3.1):

- 1) методы организации хранения информации;
- 2) методы сортировки информации;
- 3) методы поиска информации;
- 4) методы редактирования информации;
- 5) методы корректировки информации.

Более подробная классификация, характеристика и применение отдельных методов логической обработки информации рассматриваются в последующих параграфах настоящей главы.

3.1.2. Методика выбора и оценки методов логической обработки

При логической обработке информации, особенно военной, поиск записей с заданными признаками является одной из основных процедур. Но выбор метода поиска зависит от порядка расположения записей в обрабаты-

ваемом информационном массиве, т. е. от характера и степени упорядоченности массива.

По характеру упорядочения информационные массивы разделяются на следующие:

- неупорядоченные;
- упорядоченные по одному из признаков (например, по значениям одного из элементов записей);
- упорядоченные по частоте обращения к записям.

Последнее упорядочение удобно выполнять в несколько этапов. Каждый этап заключается в накоплении частоты обращения к отдельным записям за фиксированное число обращений к данному информационному массиву и в упорядочении записей в массиве по накопленным частотам обращения.

Например, накопленные частоты на предыдущих этапах учитываются с некоторым весом $0 < g \leq 1$. Если считать, что вероятности обращения к записям по времени изменяются, то $g < 1$, если они постоянны, $g = 1$. Следовательно, упорядочение по частоте обращения к записям практически сводится к упорядочению по заданному признаку, который имеет для каждой записи конкретное числовое значение.

Для практического использования можно предложить ряд разработанных методов упорядочения массивов по заданным признакам. Методы различаются по затратам машинного времени, по расходу памяти, по длине и сложности программы (рис. 3.1).

В качестве примера рассмотрим наиболее простой метод упорядочения — упорядочение (ранжировка) массива по минимуму (максимуму). Этот метод не требует дополнительного расхода памяти машины. При этом заданный признак упорядочения является одним из элементов записи.

Сущность метода. В массиве отыскивается запись с минимальным значением признака упорядочения и ставится на первое место в упорядоченном массиве. В оставшемся массиве снова отыскивается запись с минимальным значением признака и ставится на второе место. Процесс повторяется до полного упорядочения массива, т. е. $n-1$ раз при n записях в массиве.

Выполнение поиска минимума:

- переносится на рабочее поле (в ячейку) памяти

первая запись массива, запоминается место этой записи в массиве;

— последовательно просматриваются остальные записи массива. Если признак в какой-то записи имеет меньшее значение, чем для записи на рабочем поле, то на рабочее поле заносится новая запись и запоминается ее место в массиве.

После просмотра всего массива на рабочее поле заносится запись с наименьшим ($\min$) значением признака. Она ставится на первое место в массиве, а первая запись — на освободившееся место. Поиск второго $\min$ выполняется также, но со второй записи массива.

Весь процесс упорядочения по этому методу состоит из поиска $n-1$ $\min$ и выполнения $n-1$ операторов обмена для переноса каждой записи на свое место в массиве. Дополнительный расход памяти необходим на рабочее поле для одной записи при поиске минимумов и для запоминания адреса записи в массиве.

Запишем на АЛГОЛ-60 в виде процедуры ранжировки упорядочение массива, если каждая запись состоит из одного элемента, который является числом:

```
procedure range (a, n);  
  array a; integer n;  
  begin real min; integer i, j, k;  
  for k:=1 step 1 until n-1 do  
    begin min:=a[k]; j:=k;  
    for i:=k+1 step 1 until n do  
      if a[i]<min then  
        begin min:=a[i]; j:=i end;  
      a[j]:=a[k]; a[k]:=min  
    end  
  end;
```

Формальными параметрами данной процедуры являются одномерный массив a и число переменных в нем n . Анализ АЛГОЛ-программы показывает, что основными повторяющимися действиями являются условия (строка 7) и операторы присваивания (строка 8).

Наибольшие затраты машинного времени будут при упорядочении исходного массива в противоположном на-

правлении. При этом условия (сравнения) и операторы присваивания будут выполнены не более

$$N = (n-1) + (n-2) + \dots + 1 = \frac{n(n-1)}{2} \text{ раз.}$$

Если исходный массив уже упорядочен, то N раз будут выполняться только условия (сравнения).

Пусть для выполнения сравнения и обеспечения повторений цикла необходимо выполнить k операций, а операторы присваивания включают r операций. Тогда общие затраты времени будут удовлетворять неравенству

$$\frac{k}{s} \cdot \frac{n(n-1)}{2} \leq t \leq \frac{k+r}{s} \cdot \frac{n(n+1)}{2},$$

где s — среднее быстродействие ЭЦВМ (операций в секунду).

Например, при $r=7$; $k=5$, $s=10^4$ операций в секунду время на упорядочение массива из 10^3 чисел удовлетворяет неравенству $250 \text{ сек} \leq t \leq 599 \text{ сек}$, т. е. средние затраты времени составят около 7 мин.

При размещении ранжируемого массива на магнитной ленте затраты времени на упорядочение определяются в основном числом перезаписей массива на магнитную ленту.

Приведем пример приближенной оценки числа считываемых и записываемых зон в этом случае. Пусть массив из n записей размещен в m зонах МЛ, в каждой зоне которой по $p = \frac{n}{m}$ записей. При поиске каждого min можно запоминать его место в массиве, как это делается в уже описанной процедуре, а обмен для занесения очередной записи на свое место производить при следующем просмотре массива. Тогда за первый просмотр будет найден первый min . При втором просмотре он помещается на первое место в массиве и отыскивается второй min и т. д. При первом просмотре потребуется считывание всех m зон, при каждом из последующих p просмотров — считывание m зон и запись не более двух зон. В следующих p просмотрах необходимо считывать $m-1$ зону и записывать не более двух зон в каждом просмотре.

Общее число считываемых зон определяется формулой

$$\begin{aligned} M_1 &= p[m + (m-1) + \dots + 2] + m = \\ &= p \frac{(m-1)(m+2)}{2} + m. \end{aligned}$$

Число заново записываемых зон не превысит $M_2 = 2(n-p) + 1$, так как упорядочение последних p записей будет выполнено в ОЗУ, а для размещения на свои места каждой из предыдущих $n-p$ записей необходимо записывать заново не более двух зон. Число считываемых и вновь записываемых зон МЛ может оказаться значительным. Например, при $n=100$; $m=10$; $p=10$

$$M_1 = 10 \cdot \frac{9 \cdot 12}{2} + 10 = 550; \quad M_2 = 2 \cdot 90 + 1 = 181.$$

3.2. ОРГАНИЗАЦИЯ ХРАНЕНИЯ БОЛЬШИХ МАССИВОВ ДАННЫХ

3.2.1. Основные принципы хранения информации

При решении задач по обработке больших объемов информации, размещенной во внешней памяти, время обработки определяется не быстродействием вычислителя, а в основном временем обращения к внешним запоминающим устройствам. В связи с этим разработка рациональных методов организации хранения и поиска данных в машинной памяти приобретает существенное значение и позволяет в ряде случаев в десятки и сотни раз повысить эффективность обработки, т. е. косвенным путем увеличить производительность используемых ЭВМ.

Методы организации хранения и поиска являются в определенной мере общими для информации различного характера, так как рациональность использования различных методов в основном определяется структурой, формой представления информации, а также требованиями, предъявляемыми спецификой решаемых задач к системе хранения.

Основными факторами, определяющими способ организации хранения информации, являются:

- применяемый способ поиска информации;
- характеристики запоминающих устройств, применяемых для хранения информации;
- назначение, сложность организации и характер информации;

- оперативность обработки;
- частота обращения к массивам данных, хранимых во внешней памяти.

В свете перечисленных выше факторов наиболее жесткие требования к организации хранения информации предъявляются в информационно-логических задачах, сущность решения которых в общем виде сводится к разделению объектов на виды, типы, классы и т. д. в зависимости от их свойств и признаков, дополнению исходных данных и применению к полученной совокупности данных алгоритма, содержащегося в условии задачи. В целях сокращения времени обработки информации с учетом ограниченной оперативной памяти и экономии внешней памяти к способам размещения информации предъявляются следующие основные требования:

- обеспечение хранения информации, изменяющейся по своему объему и структуре;
- недопустимость хранения одних и тех же сведений дважды;
- обеспечение высокой скорости поиска и редактирования данных;
- размещение информации, используемой при одном и том же запросе, на участках внешней памяти с таким расчетом, чтобы свести к минимуму время считывания в оперативную память для обработки;
- обеспечение вывода производной информации, которая не вводилась для хранения, но получается на основе сведений, хранящихся в памяти.

В памяти ЭВМ массивы данных должны быть размещены в соответствии с требованиями на последующую их обработку. Это может быть либо последовательное размещение данных, либо их списковая структура.

При последовательном хранении информации предусматривается размещение ее по объектам и их характеристикам. Типичным примером такой организации (рис. 3.2) являются объектно-характеристические таблицы [16].

Подобные таблицы применяются тогда, когда объем информации, подлежащей хранению, известен или определен с достаточной степенью точности. В тех случаях, когда объем хранимых сведений изменяется в широких пределах, распределяя память, необходимо отводить ее по максимальным допустимым объемам сведений, под-

лежащих хранению в таблицах. При этом машинная память используется нерационально, так как средний объем сведений будет, как правило, значительно меньше максимального.

Наименование объекта	Шифр объекта	Наименование характеристик					
		X_1	X_2	X_3	$\dots$	X_{m-1}	X_m
a_1	b_1	c_{11}	c_{12}	c_{13}	$\dots$	c_{1m-1}	c_{1m}
a_2	b_2	c_{21}	c_{22}	c_{23}	$\dots$	c_{2m-1}	c_{2m}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
a_{n-1}	b_{n-1}	$c_{n-1,1}$	$c_{n-1,2}$	$c_{n-1,3}$	$\dots$	$c_{n-1,m-1}$	$c_{n-1,m}$
a_n	b_n	c_{n1}	c_{n2}	c_{n3}	$\dots$	$c_{n,m-1}$	c_{nm}

Рис. 3.2. Схема объектно-характеристической табличной формы с постоянными параметрами

Если в процессе функционирования системы обработки данных память, отводимая под массивы информации, изменяется по своему объему и структуре, то целесообразнее распределять ее по мере поступления информации. В этом случае сведения об объектах, имеющих общие признаки, располагаются не последовательно, а произвольно в различных областях памяти. Используя связи (адреса ссылок), можно объединить информацию об объектах, имеющих общий признак, в один список [18].

Классификация списков и их сравнительная оценка детально рассматриваются в ряде работ [13, 18].

Списковая организация информации позволяет в ряде случаев выполнить автоматическое перераспределение памяти в зависимости от характера поступающих сведений и обеспечивает высокую скорость поиска.

Организация хранения данных в табличной форме является по сути дела разновидностью списковой формы организации информации, так как таблицу можно рассматривать как список последовательно расположенных

элементов. Однако, учитывая существенные различия в способах организации хранения и поиска информации и разнообразные области применения, рассмотрим отдельно вопросы хранения информации в виде таблиц и в виде списков.

3.2.2. Табличный способ хранения информации

Табличный способ хранения информации предусматривает матричную структуру организации данных с последующей разверткой для хранения в линейной памяти по строкам матрицы. Развертка матрицы по столбцам в практике организации обработки информации почти не используется, за исключением отдельных алгоритмов решения транспортной задачи [18].

Объекты в таблицах располагаются в последовательности, определяемой характером обработки. Так, если система хранения обеспечивает решение заданного набора стандартных задач, то распределение объектов в таблице определяется удобством организации поиска сведений, необходимых для решения. В случае использования массива информации для получения нестандартных справок распределение информации по таблице должно выполняться в соответствии с обеспечением удобства формирования и обработки запросов и сокращения общего объема памяти, потребного для хранения информации.

Элементарную информационную совокупность, включающую шифр объекта и его характеристики, назовем записью объекта (или просто записью). Длину записи (количество занимаемых ячеек памяти) обозначим m . Количество записей, образующих массив, обозначим n .

Таблицы, в которых параметры n и m постоянны, называются таблицами с постоянными параметрами. В таких таблицах хранится информация с достаточно высоким коэффициентом стабильности [15]. Это, как правило, словари понятий, нормативные массивы постоянной информации и т. д. Редактирование информации таких таблиц выполняется в целях обновления отдельных характеристик объектов и замены отдельных записей на новые записи.

Таблицы, в которых параметры n и m переменны, принято называть таблицами с переменными параметрами. Редактирование информации подобных таблиц

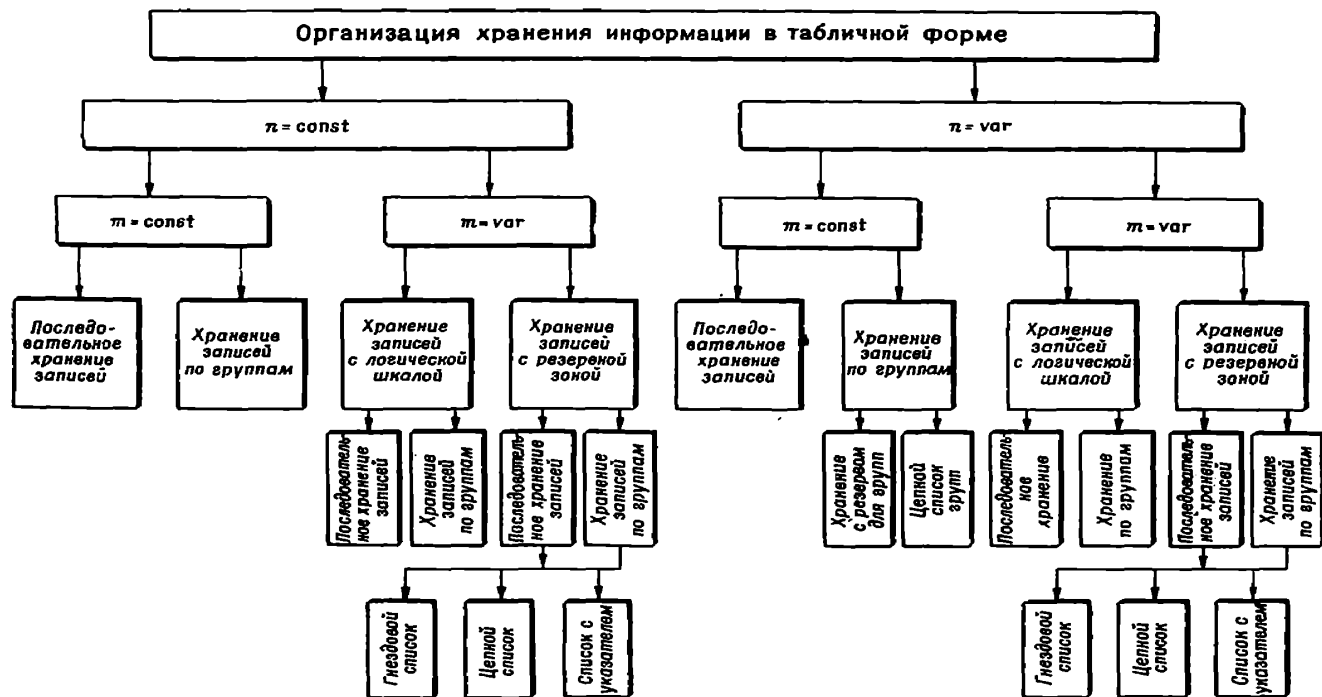


Рис. 3.3. Классификационная схема способов хранения информации

выполняется в целях изменения характеристик отдельных записей и дополнения или удаления записей.

Практические приемы организации хранения массивов информации в табличной форме представлены схемой классификации (рис. 3.3). Рассмотрим основные способы хранения больших массивов данных во внешней памяти.

Последовательное хранение записей ($n = \text{const}$, $m = \text{const}$)

Последовательное хранение записей является наиболее простым способом программной реализации. Механизм размещения информации показан на рис. 3.4. Недостаток этого способа при $n = \text{const}$ и $m = \text{const}$ очевиден: среднее время поиска записи по шифру определяется как

$$t_{\text{ср}} = \frac{n}{2}.$$

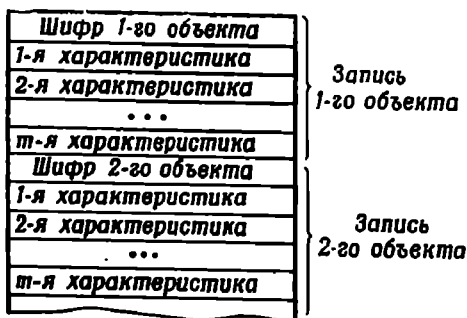


Рис. 3.4. Структура способов последовательного хранения записей

Однако в этом случае используется минимальный объем памяти на хранение записей. Подобный способ организации хранения информации в памяти может быть рекомендован для словарей понятий и нормативных массивов с низкой частотой использования.

В случае если $m = \text{var}$ и $n = \text{var}$, при последовательном хранении записей имеет место проигрыш в объеме памяти, используемом для хранения, однако при этом способе обеспечивается простая реализация алгоритма дополнения массива новыми записями, так как они размещаются в конце массива.

Хранение записей по группам

Если множество объектов может быть разбито на некоторое число подмножеств по какому-либо признаку, то возможен механизм организации хранения записей, показанный на рис. 3.5.

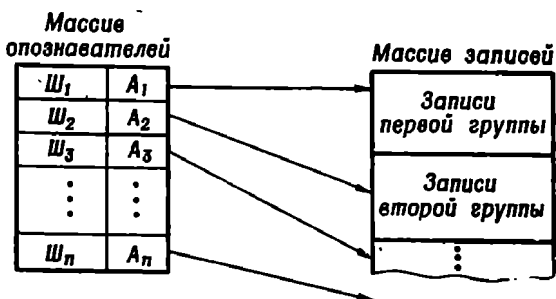


Рис. 3.5. Структура способа хранения записей по группам

Здесь $Ш_1, Ш_2, \dots, Ш_n$ — шифры групп, объединяющих некоторое количество записей, которые обладают соответствующими признаками; $A_1, A_2, \dots, A_n$ — система отсылок на местоположение первой записи соответствующих групп, размещенных во внешней памяти.

Кроме указанных компонентов массива опознавателей в нем необходимо хранить информацию о количестве слов в группе или количестве записей в группе.

Отличительной особенностью данного способа организации хранения информации является некоторое уменьшение количества ячеек внешней памяти, потребных для хранения записей за счет вынесения множества шифров $Ш_i$ в массив опознавателей. Кроме того, имеет место резкое сокращение времени на поиск заданной записи за счет уменьшения числа обращений к внешней памяти, так как поиск сканированием применяется к локализованной зоне массива записей (группе). Алгоритм первичной организации такого массива может быть представлен в виде блок-схемы (рис. 3.6) при следующих допущениях:

массив опознавателей размещается в оперативной памяти с шифрами групп, заданными в определенной последовательности;

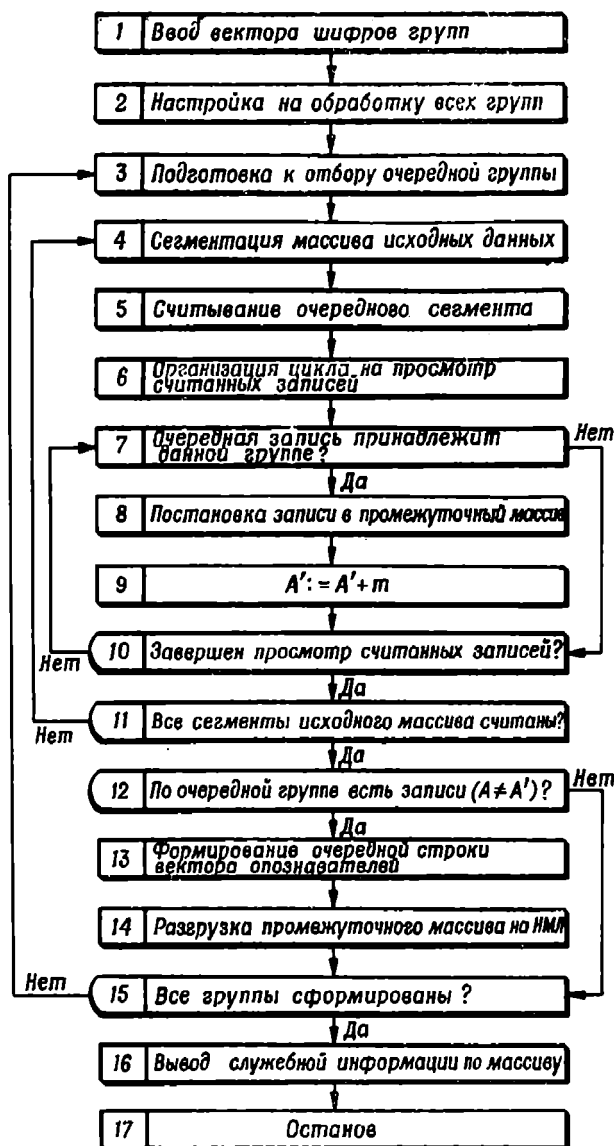


Рис. 3.8. Блок-схема алгоритма первичной организации массива по группам

исходный массив размещен на магнитной ленте в неотсортированном виде;

массив записей (итоговый массив) записывается на магнитную ленту.

Естественно, что для данного способа организации хранения массивов информации исключается дополнение их новыми записями, так как принято $n = \text{const}$.

Способ хранения записей с логической шкалой

При $m = \text{var}$ возникает необходимость уплотнения характеристик в записи с тем расчетом, чтобы общий проигрыш в использовании памяти для хранения информации сводился к минимуму.

Если в записи отсутствует свободная память между ячейками, содержащими значения характеристик, т. е. i -я характеристика может храниться в i -й ячейке записи только в том случае, когда все $(i-1)$ значения, предшествующие i -й характеристике, имеются в записи, то отсутствие значения любой характеристики приводит к сдвигу всех последующих значений к началу записи.

Определение того, к каким наименованиям характеристик относятся хранимые подряд значения, может выполняться с помощью логической шкалы. Логическая шкала представляет собой специально подобранный набор нулей и единиц. Единица в каком-либо разряде двоичной шкалы означает, что в записи имеется значение характеристики, порядковый номер наименования которой равен порядковому номеру данного разряда шкалы. Местоположение значения характеристики от начала записи определяется порядковым номером единицы в ряду других единиц логической шкалы.

Форма записи объекта с использованием логической шкалы показана на рис. 3.7.

Здесь $Ш_i$ — шифр i -го объекта; b_i — логическая шкала ($i = 1011001...1...1$); a_i^j — значение j -й характеристики i -го объекта.

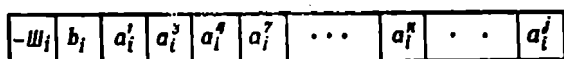


Рис. 3.7. Формальная схема записи с логической шкалой объекта

Количество разрядов в логической шкале должно быть равно (либо больше) числу наименований характеристик для объекта с их максимальным числом.

Записи размещаются в массиве последовательно одна за другой. Начало каждой записи (шифр) помечается знаком минус. Естественно, что этот способ можно применять при отсутствии отрицательных характеристик объектов.

Данный способ организации хранения записей нефиксированной длины позволяет свести к минимуму потребный для хранения объем памяти. Однако алгоритмы обработки подобных записей будут значительно усложнены за счет участков поиска начала записи и определения местоположения искомых характеристик.

Массивы записей объектов с логической шкалой во внешней памяти могут быть размещены последовательно или по группам.

В случае если алгоритмами редактирования информации предусматривается пополнение недостающих характеристик объектов, что характерно для массивов разведывательной информации, то способ организации хранения записей с логической шкалой может быть модифицирован (при $m = \text{const}$) следующим образом. По каждой из записей создается логическая шкала, в которой единица соответствует наличию показателя в записи, а нуль — его отсутствию (возможно, временному). При добавлении значений характеристик в шкалу соответствующий нуль заменяется единицей. Для алгоритма обработки информации это будет разрешением на использование данного значения характеристики.

Для случая $n = \text{var}$ и $m = \text{var}$ необходимо обеспечить пополнение записей. Хранение записей может быть последовательным, но предпочтительнее хранение по группам.

При хранении записей по группам необходимо предусмотреть резерв в каждой группе исходя из возможностей дополнения их при $n = \text{var}$.

Хранение записей с резервной зоной

При хранении записей с переменной длиной обычно поступают следующим образом. Назначается среднее значение длины записи $m_{\text{ср}}$. Если длина записей пре-

вышает $m_{\text{ср}}$, получают для хранения недостающей части характеристик резервную зону, т. е. некоторую специально отведенную область внешней памяти. Таким образом, запись объекта хранится в основной и в резервной зонах. Механизм хранения информации показан на рис. 3.8.

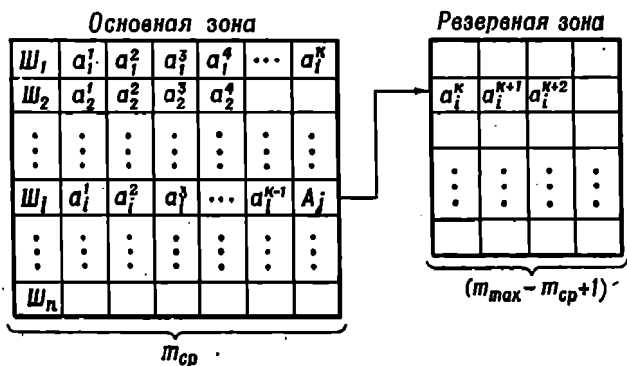


Рис. 3.8. Структура гнездового способа хранения информации с резервной зоной (при $m = \text{var}$)

Длина записи в резервной зоне определяется из выражения

$$m_{\text{max}} - m_{\text{ср}} + 1.$$

Резервная зона заполняется «излишками» записей из основного массива последовательно. Подобный способ хранения называется гнездовым списком; имеется в виду, что основная зона отождествляется с первым гнездом, а резервная — со вторым.

Модификацией способа хранения с использованием гнездового списка является цепной способ хранения информации, заключающийся в том, что запись резервной зоны назначается меньше, чем величина $m_{\text{max}} - m_{\text{ср}} + 1$. Обычно это среднее значение из «излишков». Механизм хранения информации в резервной зоне с использованием цепного списка показан на рис. 3.9.

Более существенная экономия памяти достигается при хранении информации в резервной зоне с использованием указателя. Конструкция основной зоны при этом способе аналогична той, которая принята для гнездового способа. Для размещения значений характери-

стик, не уместившихся в основной зоне, производится поиск свободного места памяти в резервной зоне. Необходимо найти последовательность свободных ячеек, содержащую не менее трех ячеек. В ячейку с наименьшим адресом на данном участке заносится указатель, позволяющий определить место хранения значений характеристик, относящихся к тому же объекту, что и указатель.

a_i^k	a_i^{k+1}	a_i^{k+2}	A_{j+1}
a_i^{k+3}	a_i^{k+4}	a_i^{k+5}	
a_i^k	a_i^{k+1}		
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Рис. 3.9. Структура способа хранения записей в резервной зоне с использованием цепного списка

Под указатель (логическую шкалу) можно отводить полную ячейку памяти, тогда число ячеек на участке будет равно разрядности данной ЭВМ. Форма размещения информации на участке резервной зоны показана на рис. 3.10. На рисунке обозначены b_i и b_l — указатели, относящиеся к i -му и l -му объектам, a_i и a_l — значения характеристик, относящихся к этим объектам. Из этого рисунка видно, что на одном и том же участке резервной зоны могут быть размещены указатели и значения характеристик, относящиеся к разным объектам. Так, например, указатель b_i предполагает использование характеристик a_i^p , a_i^{p+1} , a_i^{p+2} и a_i^{p+3} i -го объекта.

Если для объекта или их совокупности необходимо несколько участков резервной зоны, то вместо послед-

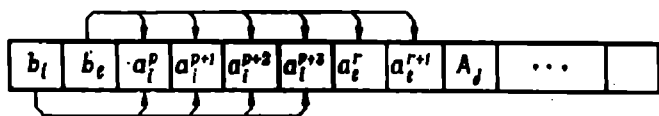


Рис. 3.10. Схема размещения информации на участке резервной зоны с указателем

него значения характеристики ставится отсылочный адрес на указатель следующего участка.

Данный способ хранения информации характеризуется более экономным расходом памяти по сравнению с предыдущими способами при условии, что на каждый указатель приходится в среднем более двух характеристик объекта. Однако необходимо отметить, что для данного способа хранения информации программа обработки более сложна, чем для предыдущих способов.

Заметим, что списковые способы хранения информации могут предусматривать внутри основной зоны как последовательное хранение записей, так и хранение по группам.

Хранение записей по группам при $n = \text{var}$

Хранение записей по группам при $n = \text{var}$ и $m = \text{const}$ применяется в том случае, когда выполняется частая коррекция записей хранимого массива в целях исключения и дополнения. Кроме того, при данном способе обычно необходимо обеспечить быстрый поиск записей как выборочно, так и последовательно для обработки.

Подобная задача может быть решена либо путем резервирования памяти внутри групп для добавления записей, либо применением цепного списка.

Первый способ не представляет особого интереса, так как не всегда удастся определить обоснованный резерв внутри групп. Однако он используется на практике сравнительно часто из-за простоты программ обработки.

Второй способ применяется в том случае, когда записи массива корректируются и, кроме того, когда алгоритмами обработки данных предусматривается как справочная, так и обычная обработка записей (получение сводных отчетов по различным разрезам, группировка, выборка и другие виды вычислительных и логических работ) при ограниченном времени выполнения вычислений.

Этот способ применяют также в том случае, когда имеет место большой разброс количества записей в отдельных группах. Предварительно устанавливается среднее количество записей на группу $n_{\text{ср}}$, характерное для большинства групп. С учетом параметров $n_{\text{ср}}$ и $m = \text{const}$ назначается стандартный объем для каждой из групп, включающий ряд вспомогательных ячеек:

идентификатор группы;
 дополнение до стандартной контрольной суммы;
 отсылку на дополнительное («младшее») хранилище записей;
 отсылку на «старшее» хранилище записей;
 фиксатор свободного поля в данном хранилище.

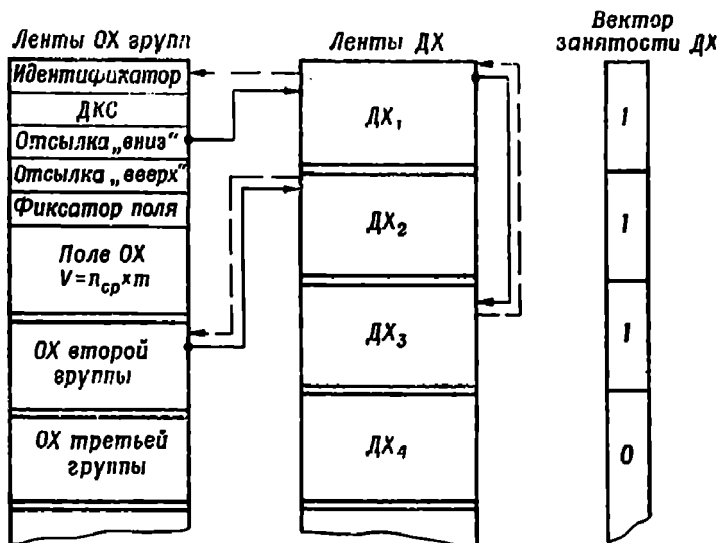


Рис. 3.11. Схема резервирования при хранении по группам в цепном списке

Двусторонняя «прошивка» отсылками обеспечивает быстрый поиск как в прямом, так и в обратном направлении в целях исключения потерь времени на перемотку магнитных лент.

Дополнительные хранилища предварительно размечаются на магнитных лентах и резервируются за группами при переполнении основных хранилищ. Занятость дополнительных хранилищ фиксируется в специальном векторе занятости. Механизм организации хранения показан на рис. 3.11.

На данном рисунке обозначены: ОХ — основные хранилища групп; ДХ — дополнительные хранилища. Изображенный на рисунке пример показывает, что основное хранилище первой группы резервирует первое дополнительное хранилище ДХ₁, которое в свою очередь

резервирует при условии занятости $ДХ_2$ под информацию второй группы следующее по порядку дополнительное хранилище $ДХ_3$. В векторе занятости первым трем занятым дополнительным хранилищами соответствуют единичные состояния элементов.

Объемы основных и дополнительных хранилищ назначаются из расчета минимума времени на поиск за счет «пробежки» лент при обращениях.

Алгоритм работы с хранилищами в данном случае оформлен в виде стандартной процедуры, обеспечивающей загрузку (выгрузку) основных хранилищ в соответствии с шифрами групп с последующим анализом наличия отсылок «вниз» на дополнительные хранилища.

3.2.3. Хранение информации в виде списков

Идея списочной организации информации основана на разбиении массива на последовательность частей и включении в каждую из частей ссылок на следующую и (или) предыдущую части [18]. Под ссылкой на часть понимается информация о местоположении этой части.

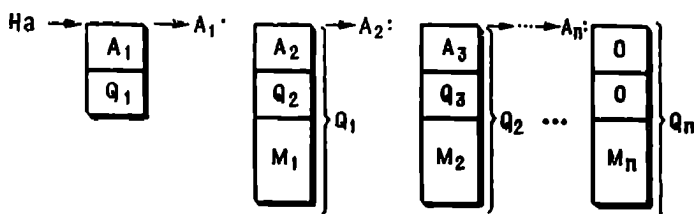


Рис. 3.12. Схема списочной организации с ссылками вперед

Нулевая ссылка рассматривается как отсутствие продолжения. Схематически списочная организация с ссылками только вперед изображена на рис. 3.12. Здесь массив M состоит из частей $M_1, M_2, \dots, M_n$, HA — адрес информации о первой части массива и его длине.

Под информацией о части массива M_k понимается адрес A_k ее начала и длина V_k с учетом дополнительной информации о следующей части.

Пусть необходимо организовать последовательную выборку из массива (слово за словом). Будем называть текущими выбранное слово и содержащую его часть массива. AT и VT — переменные, значения которых рав-

ны соответственно адресу начала текущей части массива и его длине; k — переменная, значение которой есть номер текущего слова в текущей части массива, и P — переменная, которой присваивается значение текущего слова массива. Выборка очередного слова массива описывается алгоритмом:

```
...S : if  $AT = 0$  then goto — конец массива;  
        if  $k \geq VT$  then  
begin  $k := 2$ ;  $VT := M[AT + 1]$ ;  $AT := M[AT]$ ; goto S  
    end;  $P := M[AT + k]$ ;  $k := k + 1$ ;
```

Для выборки первого элемента необходимо предварительно выполнить операторы:

$$AT := HA; VT := 2; k := 2.$$

Таким образом, значение выражения $M[E]$ является содержимым ячейки памяти, адрес которой есть значение выражения E .

Возникновение массива, состоящего из отдельных частей, можно представить себе следующим образом. В процессе автоматической обработки информации появляются некоторые массивы U , V , T и т. д. Будем считать, что каждый из массивов упорядочен и следующее по порядку слово массива появляется только после предыдущего. Для временного накопления появляющихся слов массивов выделим области оперативной памяти: BU , BV , BT и т. д.; будем называть их сегментами соответствующих массивов.

В процессе заполнения любого из сегментов могут возникать условия, делающие необходимым или желательным очистку сегмента, т. е. перенесение накопленной информации из сегмента в другое место памяти для хранения. Такими условиями могут быть полная загрузка сегмента или, например, сообщение о временном или полном прекращении появления новых слов рассматриваемого массива и желательность использования сегмента для других целей.

Предположим далее, что, начиная с ячейки с адресом F , в нашем распоряжении имеется свободная память и что для каждого массива введены переменные: UA , VA , TA , ..., значения которых равны адресам ссылок на первую часть соответствующего массива или ну-

лю до переноса части массива из соответствующего сегмента.

Кроме того, введены переменные: $UK, VK, TK, \dots$, текущие значения которых равны числу слов массивов, накопленных в соответствующих сегментах: $BU, BV, BT, \dots$. Начальные значения этих переменных будем считать нулевыми.

Под $BU[E]$ понимаем слово, порядковый номер которого в сегменте BU равен значению индексного выражения E . Освобождение любого сегмента может быть выполнено оператором процедуры (применительно к сегменту BU):

$STORE (UA, UK, BU, F),$

где процедура $STORE$ определяется описанием:

```
procedure STORE (A, K, B, F); integer A, K, F; array B;  
  begin integer C, J, H, L;  
    if K = 0 then begin if A = 0 then  
begin A := F; M[A] := M[A + 1] := 0; F := F + 2 end;  
      C := A; S: if M[C] ≠ 0 then  
        begin C := M[C]; goto S end;  
      H := M[C] := F; L := M[C + 1] := K + 2;  
      F := F + L; M[H] := M[H + 1] := 0;  
      for J := 1 step 1 until K do  
        M[H + J + 1] := B[J]; K := 0  
      end  
    end;
```

Описанная процедура содержит многократно повторяемый для поиска адреса последней ссылки оператор с меткой S , который может быть исключен, если в число параметров процедуры освобождения сегмента ввести переменную, текущее значение которой есть адрес последней ссылки. Присваивая этой переменной новое значение в процедуре очистки сегмента, можно обеспечить подготовку к новому обращению.

Для изложенной организации массива выборка заданного элемента $U[1]$ может быть выполнена оператором процедуры

$LOAD (T, UA, J),$

где процедура *LOAD* реализует оператор присваивания

$$T := U[J]$$

и описывается следующим образом:

```
procedure LOAD (T, A, J); value A, J; integer A, J;  
begin integer L;  
  if  $J \leq 0$  then goto ошибка индекса;  
  L := 2; S: if A = 0 then goto элемента нет;  
    J := J + 2 - L; if  $J > 0$  then  
begin L := M[A + 1]; A := M[A]; goto S end;  
  T := M[A + J + L - 1]  
end;
```

Процедура, выполняющая оператор присваивания $U[J] := T$, отличается от приведенной выше лишь последним оператором присваивания, который должен быть заменен оператором

$$M[A + J + L - 1] := T.$$

Если заменить этот оператор условным оператором:

```
if B then  $M[A + J + L - 1] := T$  else  $T := M[A + J + L - 1]$ ,
```

ввести в список формальных параметров процедуры параметр *B*, внеся его в список значений с *value* и специфицируя как **Boolean**, получим процедуру, выполняющую «выборку» при значении $B = \text{false}$ и «занесение» при true .

В эту процедуру полезно включать формальные параметры и соответствующие операторы для получения значения адреса начала и длины части массива, в которой находится искомый элемент, а также значение порядкового номера этого элемента в данной части.

Списочная организация позволяет легко присоединить части одного массива к другому как в конце первого, так и между заданными частями первого. Предположим, что к массиву *U* требуется присоединить группу следующих друг за другом частей массива *V*, исключив эту группу из массива *V*, как это показано на рис 3.13.

Для исключения и присоединения частей списка достаточно выполнить пересылку в ячейки A_n и A_{n+1} содержимого ячеек C_n и C_{n+1} . В две последние ячейки необходимо переслать содержимое ячеек C_{n+r} и C_{n+r+1} ,

заполнив ячейки C_{k+r} и C_{k+r+1} нулями. Эти действия, включая и поиск необходимых адресов, можно описать следующими операторами:

$A := VA; R : \text{if } M[A] \neq 0 \text{ then begin } A := M[A]; \text{ goto } R \text{ end};$
 $C := VA; S : \text{if } \neg P(C) \text{ then begin } C := M[C]; \text{ goto } S \text{ end};$
 $D := C; T : \text{if } \neg Q(D) \text{ then begin } D := M[D]; \text{ goto } T \text{ end};$
 $M[A] := M[C]; M[A + 1] := M[C + 1];$
 $M[C] := M[D]; M[C + 1] := M[D + 1];$
 $M[D] := M[D + 1] := 0;$

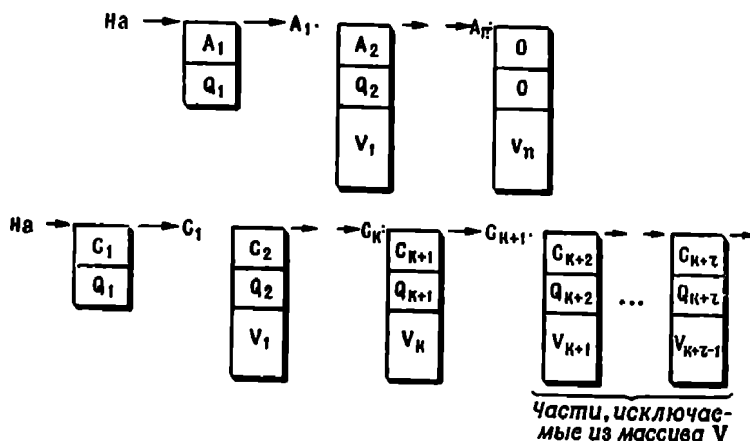


Рис. 3.13. Схема исключения части списка

Здесь P и Q — идентификаторы логических процедур-функций, принимающих значение **true** когда значение фактического параметра в рассматриваемом случае оказывается равным соответственно C_k и C_{k+r} . Можно сказать, что процедуры P и Q распознают начало и конец последовательности, исключаемой из массива V .

Рассматриваемая списочная организация с ссылкой на следующую часть удобна в том случае, когда основная работа с массивом выполняется в режиме просмотра от начала к концу. Но при подобной организации списка нельзя непосредственно перейти к предыдущей части, так как нет соответствующей информации. В целях ликвидации этого недостатка в дополнительную информацию каждой части списка необходимо ввести

отсылочный адрес и объем предыдущей части, что позволит проводить просмотр массива с конца так же просто, как и с начала. Дополнительное удобство (обратный список) достигается ценой увеличения общего объема информации и целесообразность его введения в общем случае не очевидна.

В соответствии с тем, что для входа в прямой список имеется так называемая нулевая часть, вводится понятие заключительной части, обеспечивающей просмотр списков в обратном направлении. Так же как и нулевая

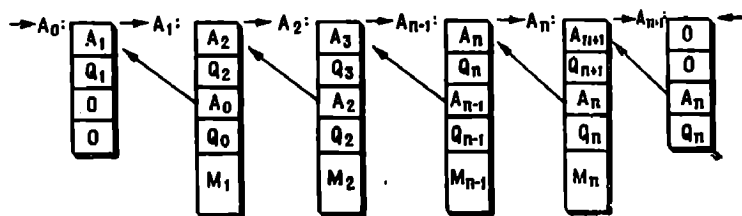


Рис. 3.14. Схема списочной организации с прямыми и обратными ссылками

часть при ссылках только вперед, нулевая и заключительная части при ссылках в обе стороны упрощают алгоритмы включения и исключения частей, избавляя от анализа краевых ситуаций. Таким образом, предполагаем, что кроме *НА* с массивом связана переменная *КА*, значение которой есть адрес начала крайней правой части списка. Общий вид списковой организации с прямыми и обратными ссылками показан на рис. 3.14.

В целях сопоставления объема работ при отсутствии и наличии обратных ссылок рассмотрим процедуру исключения из массива текущей части, адрес начала которой есть значение переменной *УС*.

При отсутствии ссылок назад такая процедура может быть задана следующим описанием:

```

procedure DELETE (A, C);
  value A, C; integer A, C;
  begin S : if M[A]  $\neq$  C then
    begin A := M[A]; goto S end;
  M[A] := M[C]; M[A + 1] := M[C + 1]
  end;
  
```


Исключение производится оператором процедуры *DELETE (UA, UC)*.

При наличии обратных ссылок, организованных, как показано на рис. 3.14, рассматриваемое исключение реализуется следующей группой операторов:

$$\begin{aligned}M[M[UC + 2]] &= M[UC]; \\ M[M[UC] + 1] &= M[UC + 1]; \\ M[M[UC] + 2] &= M[UC + 2]; \\ M[M[UC] + 3] &= M[UC + 3];\end{aligned}$$

Первая пара операторов задает перенос ссылки вперед в предыдущую часть на место ссылки на исключаемую часть, а вторая пара — перенос ссылки назад в следующую часть.

Вставка новой части вслед за частью, начинающейся с адреса *C*, исходя из предположения, что в ее начале имеются четыре ячейки для ссылок и имеются ссылки вперед и назад, выполняется оператором:

$$\begin{aligned}\text{begin } M[F] &= M[C]; M[F + 1] = M[C + 1]; \\ M[F + 2] &= C; \\ M[F + 3] &= M[M[C] + 3]; M[C + 1] = \\ &= M[M[C] + 3] = L; \\ M[C] &= M[M[C] + 2] = F; \\ \text{end};\end{aligned}$$

Здесь *F* и *L* соответственно адрес начала и длина вставки. Случай присоединения новой части в конец списка является частным случаем рассмотренного допущения и осуществляется этим же оператором при условии, что *C* совпадает с *M[KA + 2]*.

При рассмотрении списковой организации массивов предполагалось, что части могут иметь различную длину. Это обстоятельство вынуждало иметь в ссылке информацию о длине и увеличивало объем ссылок, однако давало определенные преимущества и свободу при организации массивов и работе с ними.

Основные затруднения при работе с частями массива различной длины возникают при желании использовать освободившуюся память после того, как весь массив или его части больше не нужны в памяти машины. Различие длины освобождаемых и требующихся участ-

ков памяти приводит к измельчению частей и к необходимости проведения в некоторый момент реорганизации массива и перераспределения памяти. Поэтому в ряде случаев оказывается целесообразным остановиться на фиксированной длине частей.

Все приведенные выше алгоритмы с небольшими изменениями, касающимися учета длины частей, пригодны и для рассматриваемого случая. Особого внимания заслуживают нулевая и заключительная части. Их нестандартная длина приводит к тому, что для их размещения необходимо отводить отдельный участок памяти, чтобы не потерять преимущества, предоставляемые наличием фиксированной длины частей.

Так, в алгоритме процедуры *STORE* при $A=0$ место для начальной ссылки необходимо выбрать в соответствии со значением специального указателя (дополнительного параметра процедуры) первой свободной ячейки в области памяти, отведенной под начальные списки. В этой же процедуре целесообразно изменить алгоритм выделения места под новую часть. Можно предположить, что вся свободная память организована как массив свободных частей с ссылками вперед и выделение памяти под новую часть некоторого массива информации сводится к исключению части из массива свободной памяти и присоединения исключенной части к массиву информации для заполнения этой части новой информацией.

Любое исключение части или цепочки частей из некоторого массива можно сопровождать на практике присоединением исключенных частей к массиву свободных частей. Очевидно, что при исключении части из массива свободных частей и включении в него новой можно всегда начинать с первой части этого массива. Поэтому для массива свободных частей достаточно иметь переменную, значение которой есть адрес первой части, и не рассматривать значение адреса этой переменной, что несколько упрощает описание алгоритма исключения и включения.

В алгоритме процедуры *STORE* при условии, что F есть адрес начала первой части массива свободных частей, продвижение по списку можно осуществлять оператором присваивания вместо оператора $F := F + L$.

Аналогично можно поступить с совокупностью ячеек, предназначенных для нулевых частей. Эту совокупность можно организовать как массив свободных ячеек, в каждой из которых записан адрес следующей. Исключая из этого массива первую ячейку, используем ее под нулевую часть нового массива информации. При уничтожении массива эту ячейку возвращаем в массив свободных ячеек, помещая в нее соответствующую ссылку. Признаком завершения массива свободных ячеек и массива свободных частей может служить равенство нулю соответствующей переменной, задающей начало свободной цепочки.

В схеме освобождения сегментов и накопления массивов для случая работы с частями постоянной длины необходимо решить задачу освобождения неполного сегмента при завершении обработки. Для этого можно условиться о признаке конца информации в сегменте и считать, что это некоторый код, например все единицы двоичного машинного слова.

Рассмотренные выше методы списковой организации ориентированы на использование массивов в режиме последовательного просмотра. В том случае, когда требуется доступ к различным частям массива в произвольном, случайном порядке, линейная структура ссылок приводит к большим потерям времени при поиске заданной информации. Для ускорения процесса поиска в этом случае организуется дерево ссылок. Основой для такой организации является наличие некоторого транзитивного отношения упорядоченности $P < Q$ между частями информации. Таким отношением может быть, например, лексикографическое упорядочение имен (идентификаторов) частей, если с каждой из них сопоставлено такое имя, упорядочение по адресам расположения в памяти и т. д.

Пусть имеет место такое отношение упорядоченности и пусть $M_1, M_2, \dots, M_n$ — части, размещаемые в соответствии с условием

$$2^{m-1} \leq n < 2^m.$$

Иначе: $M_j < M_k$ тогда и только тогда, когда $j < k$. Свяжем с частью M_k для четных k ссылки на части $M_{G(k)}$ и $M_{L(k)}$, где $G(k)$ и $L(k)$ определяются следующим образом.

Пусть

$$k = \sum_{j=0}^{m-1} b_j 2^j,$$

где $b_j = 0, 1$.

Тогда

$$G(k) = k + 2^{S-1} \text{ и } L(k) = k - 2^{S-1}$$

при

$$S = \min \{j \mid b_j \neq 0\}.$$

Ссылка будет нулевой, если $G(k) > n$.

Предположим, что имея адрес начала части, можно установить, предшествует ли эта часть искомой части, следует ли она за ней или совпадает с искомой. Это означает, что, следуя по ссылкам и определив адрес очередной части, можно определить, закончен ли поиск или по какой из двух ссылок на $M_{G(k)}$ или $M_{L(k)}$ — следует идти дальше. Ссылкой на $M_{G(k)}$ можно пользоваться, если искомая часть следует за найденной, и ссылкой на $M_{L(k)}$, если искомая часть предшествует найденной.

Объем работы по поиску заданной части определяется тем, что число просмотренных ссылок не превосходит $\text{C} \lg_2 n$, если начинать поиск по адресу начала части M_2^{m-1} . Итак, если очередная часть содержит отсылку вверх и вниз, то определение адреса C начала части по заданному I осуществляется следующим алгоритмом:

```
begin C := A; S : if C = 0 then goto части нет;  
           if M[C] = I then goto R;  
           if M[C] < I then C := M[C + 1] else  
           C := M[C + 2]; goto S; R : end;
```

Здесь A — адрес части M_2^{m-1} .

Пример связывающих ссылок для случая шести частей исходного массива показан на рис. 3.15.

Одним из существенных недостатков подобной организации массивов является необходимость неизбежной перестройки системы ссылок при дополнении списка новой частью. Однако существуют способы организации ссылок, позволяющих в случае дополнений проводить только локальную перестройку.

В рассматриваемом варианте организации списка

предполагалось хранение ссылок вместе с частями информации.

При хранении массива во внешней памяти его оглавление, т. е. информация о местоположении частей, желательно иметь для анализа в оперативной памяти в целях организации быстрого поиска частей.

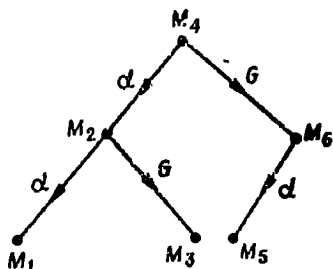


Рис. 3.15. Граф ссылок
(при $n = 6$)

3.2.4. Хранение массивов в библиотеке

Функционирование автоматизированных систем управления связано с обработкой, накоплением и хранением больших объемов ин-

формации [7]. Эта информация объединяет нормативно-справочные массивы, массивы, содержащие информацию о предшествующих этапах функционирования системы, массивы оперативных данных о текущем состоянии и программы решения задач системы. Вся эта информация является неотъемлемой частью информационно-вычислительного комплекса и хранится в его информационной библиотеке.

При рассмотрении вопросов организации массивов отмечалось, что выбор способа определяется в основном режимами использования массивов, под которыми понимаются справочные, поисковые или корректирующие режимы работы.

При справочном режиме работы с массивом предполагается выбор записей по заданному признаку для их последующей обработки. Этот режим характерен как для информационных, так и для вычислительных задач.

Поисковый режим работы с массивом предполагает выбор записи или группы записей, обладающих заданным признаком, либо установление факта отсутствия таких записей.

При корректирующем режиме работы с массивом выполняется редактирование записей по заданным признакам, включающее изменение отдельных показателей записи, исключение и (или) добавление записей.

В каждом из этих режимов общее время решения

поставленной задачи определяется порядком организации массива. Так, если массив организован по признаку, который является основным для обращения в конкретном режиме, то время решения задачи будет сведено к минимуму. Если хранение массива выполнено по другому признаку, то увеличивается как время ожидания результата, так и стоимость выполнения обращения.

Одним из путей сокращения величин этих показателей является организация хранения нескольких дублей массива, каждый из которых упорядочен по определенному признаку в строгом соответствии с потребностями решаемых задач. Предполагается хранение этих дублей в информационной библиотеке с указанием соответствующих признаков организации, наличие разработанной методики обновления массивов после коррекций (приведение в соответствие) и разработка специального программного обеспечения ведения архивного хозяйства библиотеки.

Анализ целесообразности дублирования массивов, выполненный в работе [7], показывает, что с увеличением количества массивов-дублей среднее время ожидания результата при справочном и поисковом режимах работы монотонно убывает и достигает наименьшего значения при условии, что количество дублей соответствует числу признаков в записи. Эксплуатационная стоимость библиотеки достигает минимального значения при хранении массивов с двумя-тремя дублями. Оптимальное количество массивов-дублей с точки зрения хранения и обработки массивов находится между n_v и m , где n_v — число массивов-дублей, полученное при минимальной эксплуатационной стоимости; m — число признаков в записи массива.

Для массивов, хранящихся на магнитной ленте, в библиотеке по каждому из дублей должно храниться не менее двух копий.

3.2.5. Организация массивов информации в ЭЦВМ «Минск-32»

В системе программирования «Минск-32» (М-32) предполагается, что все массивы информации (программы на исходных языках и языке загрузки, наборы отла-

дочных и выходных данных) организуются стандартным образом. Каждый массив представляется на носителях отдельными частями, называемыми информационными блоками (ИБ). В качестве ИБ на магнитных лентах применяется зона, на перфокартах — карта, на перфолентах — символ, в печатном сообщении — печатный лист.

Предполагается, что элементарной составляющей ИБ является запись, представляющая некоторую логически завершенную последовательность слов (символов). В содержании и свойствах записи отражаются содержание и свойства массива информации.

Запись на различных носителях может быть представлена целым числом карт (на перфокартах), любым количеством символов (на перфолентах) и одной или частью зоны на магнитной ленте. По количеству слов, составляющих запись, допускаются записи фиксированной, переменной или неопределенной длины. В массиве могут содержаться записи только одного из допустимых типов.

В массивах записей фиксированной длины количество слов, составляющих любую запись массива, одинаково для каждой записи. Скорость пересылки и обработки записей фиксированной длины значительно выше скоростей обработки и пересылки других типов записей. Отображение массива записей этого типа на магнитной ленте производится зонами фиксированной длины, кратной длине записи, на перфокартах — количеством карт, кратных целому числу карт одной записи. Для перфолент этот тип записи применяется сравнительно редко.

В массивах, содержащих записи различной длины, каждая запись должна содержать информацию о своей длине. На магнитных лентах массив таких записей представляется зонами переменной длины, содержащими одну или несколько записей. На других носителях записи переменной длины представляются переменным количеством карт или символов. Этот тип записи удобен для обработки и применим для всех носителей, но формируется только программно.

Размер записи неопределенной длины ограничивается специальным символом-разделителем. Обработка таких записей трудоемка, но это наиболее приемлемый тип записи при ручной подготовке данных на перфолентах.

В целях разграничения массивов, их идентификации и определения параметров каждый массив снабжается справочной информацией, которая сосредоточивается в контрольном блоке. Различают начальный и конечный контрольные блоки массива. Начальный контрольный блок (НБ) записывается перед первым информационным блоком массива и имеет следующую структуру (рис. 3.16).

	0	6	13	20	27	34	36
1	*	*	*	Н	Б		
2	Наименование			массива			
3				Срок год-			
4	ности						
5	Д а т а			о б р а -			
6	з о в а н и я			Признак		Признак	
7	Раздели-			обмена		типа	
8	записей			записи			
	Размер			записи			
	Резерв						
16	Резерв						

Рис. 3.16. Структура НБ массива

Информация НБ представлена 80 символами кода ГОСТ 10859—64, размещаемыми в 16 ячейках памяти. Первые пять символов НБ содержат шифр контрольного блока (** * НБ). Следующие пять произвольных символов (6—10-й) содержат наименование массива. Символами 13—20-й задается срок годности массива (четыре символа задают год, два следующих — месяц и два последних — день). Символы с 21-го по 28-й содержат информацию, указывающую год, месяц и день образования массива.

29-й символ содержит единицу, если при записи и чтении массива используются все 37 двоичных разрядов ячейки, т. е. обмен информацией производится по 6 сим-

волов в слове. Если обмен производится по 5 символов в слове, то 29-й разряд содержит символ нуля.

30-й символ содержит признак типа записи. Если массив содержит записи фиксированной длины, то 30-й символ содержит нуль. Если массив содержит записи переменной длины, то 30-й символ содержит единицу.

31-й символ содержит код разделителя записей в массиве или нуль при отсутствии разделителя.

	0	6	13	20	27	34	36
1	*	*	*	К	Б		
2	На	им	е	н	о	в	а
3	н	о	м	е	н	о	в
4	а	н	и	м	е	н	о
5	в	а	н	и	м	е	н
6	о	в	а	н	и	м	е
7	н	о	м	е	н	о	в
8	а	н	и	м	е	н	о
9	в	а	н	и	м	е	н
10	о	в	а	н	и	м	е
11	н	о	м	е	н	о	в
12	а	н	и	м	е	н	о
13	в	а	н	и	м	е	н
14	о	в	а	н	и	м	е
15	н	о	м	е	н	о	в
16	а	н	и	м	е	н	о
17	в	а	н	и	м	е	н
18	о	в	а	н	и	м	е
19	н	о	м	е	н	о	в
20	а	н	и	м	е	н	о
21	в	а	н	и	м	е	н
22	о	в	а	н	и	м	е
23	н	о	м	е	н	о	в
24	а	н	и	м	е	н	о
25	в	а	н	и	м	е	н
26	о	в	а	н	и	м	е
27	н	о	м	е	н	о	в
28	а	н	и	м	е	н	о
29	в	а	н	и	м	е	н
30	о	в	а	н	и	м	е
31	н	о	м	е	н	о	в
32	а	н	и	м	е	н	о
33	в	а	н	и	м	е	н
34	о	в	а	н	и	м	е
35	н	о	м	е	н	о	в
36	а	н	и	м	е	н	о
37	в	а	н	и	м	е	н
38	о	в	а	н	и	м	е
39	н	о	м	е	н	о	в
40	а	н	и	м	е	н	о
41	в	а	н	и	м	е	н
42	о	в	а	н	и	м	е
43	н	о	м	е	н	о	в
44	а	н	и	м	е	н	о
45	в	а	н	и	м	е	н
46	о	в	а	н	и	м	е
47	н	о	м	е	н	о	в
48	а	н	и	м	е	н	о
49	в	а	н	и	м	е	н
50	о	в	а	н	и	м	е
51	н	о	м	е	н	о	в
52	а	н	и	м	е	н	о
53	в	а	н	и	м	е	н
54	о	в	а	н	и	м	е
55	н	о	м	е	н	о	в
56	а	н	и	м	е	н	о
57	в	а	н	и	м	е	н
58	о	в	а	н	и	м	е
59	н	о	м	е	н	о	в
60	а	н	и	м	е	н	о
61	в	а	н	и	м	е	н
62	о	в	а	н	и	м	е
63	н	о	м	е	н	о	в
64	а	н	и	м	е	н	о
65	в	а	н	и	м	е	н
66	о	в	а	н	и	м	е
67	н	о	м	е	н	о	в
68	а	н	и	м	е	н	о
69	в	а	н	и	м	е	н
70	о	в	а	н	и	м	е
71	н	о	м	е	н	о	в
72	а	н	и	м	е	н	о
73	в	а	н	и	м	е	н
74	о	в	а	н	и	м	е
75	н	о	м	е	н	о	в
76	а	н	и	м	е	н	о
77	в	а	н	и	м	е	н
78	о	в	а	н	и	м	е
79	н	о	м	е	н	о	в
80	а	н	и	м	е	н	о
81	в	а	н	и	м	е	н
82	о	в	а	н	и	м	е
83	н	о	м	е	н	о	в
84	а	н	и	м	е	н	о
85	в	а	н	и	м	е	н
86	о	в	а	н	и	м	е
87	н	о	м	е	н	о	в
88	а	н	и	м	е	н	о
89	в	а	н	и	м	е	н
90	о	в	а	н	и	м	е
91	н	о	м	е	н	о	в
92	а	н	и	м	е	н	о
93	в	а	н	и	м	е	н
94	о	в	а	н	и	м	е
95	н	о	м	е	н	о	в
96	а	н	и	м	е	н	о
97	в	а	н	и	м	е	н
98	о	в	а	н	и	м	е
99	н	о	м	е	н	о	в
100	а	н	и	м	е	н	о
101	в	а	н	и	м	е	н
102	о	в	а	н	и	м	е
103	н	о	м	е	н	о	в
104	а	н	и	м	е	н	о
105	в	а	н	и	м	е	н
106	о	в	а	н	и	м	е
107	н	о	м	е	н	о	в
108	а	н	и	м	е	н	о
109	в	а	н	и	м	е	н
110	о	в	а	н	и	м	е
111	н	о	м	е	н	о	в
112	а	н	и	м	е	н	о
113	в	а	н	и	м	е	н
114	о	в	а	н	и	м	е
115	н	о	м	е	н	о	в
116	а	н	и	м	е	н	о
117	в	а	н	и	м	е	н
118	о	в	а	н	и	м	е
119	н	о	м	е	н	о	в
120	а	н	и	м	е	н	о
121	в	а	н	и	м	е	н
122	о	в	а	н	и	м	е
123	н	о	м	е	н	о	в
124	а	н	и	м	е	н	о
125	в	а	н	и	м	е	н
126	о	в	а	н	и	м	е
127	н	о	м	е	н	о	в
128	а	н	и	м	е	н	о
129	в	а	н	и	м	е	н
130	о	в	а	н	и	м	е
131	н	о	м	е	н	о	в
132	а	н	и	м	е	н	о
133	в	а	н	и	м	е	н
134	о	в	а	н	и	м	е
135	н	о	м	е	н	о	в
136	а	н	и	м	е	н	о
137	в	а	н	и	м	е	н
138	о	в	а	н	и	м	е
139	н	о	м	е	н	о	в
140	а	н	и	м	е	н	о
141	в	а	н	и	м	е	н
142	о	в	а	н	и	м	е
143	н	о	м	е	н	о	в
144	а	н	и	м	е	н	о
145	в	а	н	и	м	е	н
146	о	в	а	н	и	м	е
147	н	о	м	е	н	о	в
148	а	н	и	м	е	н	о
149	в	а	н	и	м	е	н
150	о	в	а	н	и	м	е
151	н	о	м	е	н	о	в
152	а	н	и	м	е	н	о
153	в	а	н	и	м	е	н
154	о	в	а	н	и	м	е
155	н	о	м	е	н	о	в
156	а	н	и	м	е	н	о
157	в	а	н	и	м	е	н
158	о	в	а	н	и	м	е
159	н	о	м	е	н	о	в
160	а	н	и	м	е	н	о
161	в	а	н	и	м	е	н
162	о	в	а	н	и	м	е
163	н	о	м	е	н	о	в
164	а	н	и	м	е	н	о
165	в	а	н	и	м	е	н
166	о	в	а	н	и	м	е
167	н	о	м	е	н	о	в
168	а	н	и	м	е	н	о
169	в	а	н	и	м	е	н
170	о	в	а	н	и	м	е
171	н	о	м	е	н	о	в
172	а	н	и	м	е	н	о
173	в	а	н	и	м	е	н
174	о	в	а	н	и	м	е
175	н	о	м	е	н	о	в
176	а	н	и	м	е	н	о
177	в	а	н	и	м	е	н
178	о	в	а	н	и	м	е
179	н	о	м	е	н	о	в
180	а	н	и	м	е	н	о
181	в	а	н	и	м	е	н
182	о	в	а	н	и	м	е
183	н	о	м	е	н	о	в
184	а	н	и	м	е	н	о
185	в	а	н	и	м	е	н
186	о	в	а	н	и	м	е
187	н	о	м	е	н	о	в
188	а	н	и	м	е	н	о
189	в	а	н	и	м	е	н
190	о	в	а	н	и	м	е
191	н	о	м	е	н	о	в
192	а	н	и	м	е	н	о
193	в	а	н	и	м	е	н
194	о	в	а	н	и	м	е
195	н	о	м	е	н	о	в
196	а	н	и	м	е	н	о
197	в	а	н	и	м	е	н
198	о	в	а	н	и	м	е
199	н	о	м	е	н	о	в
200	а	н	и	м	е	н	о
201	в	а	н	и	м	е	н
202	о	в	а	н	и	м	е
203	н	о	м	е	н	о	в
204	а	н	и	м	е	н	о
205	в	а	н	и	м	е	н
206	о	в	а	н	и	м	е
207	н	о	м	е	н	о	в
208	а	н	и	м	е	н	о
209	в	а	н	и	м	е	н
210	о	в	а	н	и	м	е
211	н	о	м	е	н	о	в
212	а	н	и	м	е	н	о
213	в	а	н	и	м	е	н
214	о	в	а	н	и	м	е
215	н	о	м	е	н	о	в
216	а	н	и	м	е	н	о
217	в	а	н	и	м	е	н
218	о	в	а	н	и	м	е

Организация массивов данных на магнитных лентах

При размещении массивов на магнитных лентах возникает необходимость введения специальных контрольных блоков катушек ленты: начало катушки (НК) и конец катушки (КК). Эти блоки, состоящие из 16 ячеек, позволяют отличать одну катушку от другой, а также ограничивать информацию на ленте.

Структура НК и КК показана соответственно на рис. 3.18 и 3.19.

	0	6	13	20	27	34	36
1	*	*	*	Н	К		
	1	2	3	4	5		
2	Наименование катушки ленты						
	6	7	8	9	10		
3		Р е з е р в					
	11	12	13	14	15		
⋮							
16		Р е з е р в					
	76	77	78	79	80		

Рис. 3.18. Структура НК

	0	6	13	20	27	34	36
1	*	*	*	К	К		
	1	2	3	4	5		
2							
	6	7	8	9	10		
3		Р е з е р в					
	11	12	13	14	15		
⋮							
16		Р е з е р в					
	76	77	78	79	80		

Рис. 3.19. Структура КК

Если на магнитную ленту больше нельзя записывать информацию из-за некачественной ленты или близости конца ленты, то символы 6—10-й КК дублируются символами 1—5-й (**КК). Если же запись информации на данную катушку ленты еще возможна, то в 6—10-м символах записываются пробелы.

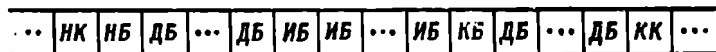


Рис. 3.20. Схема расположения информации одного массива на одной катушке МЛ

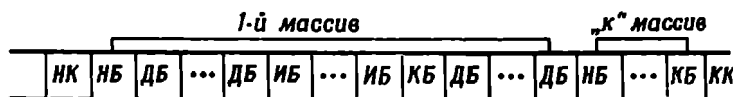


Рис. 3.21. Схема расположения информации нескольких (К) массивов на одной катушке МЛ

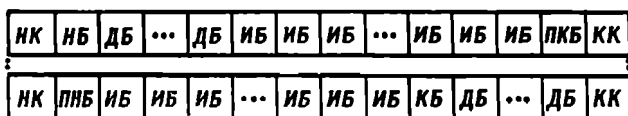


Рис. 3.22. Схема расположения информации одного массива на нескольких катушках МЛ

Для массива, размещаемого на ленте, допускается введение дополнительных контрольных блоков (ДБ) непосредственно за НБ и КБ. Блоками ДБ программист может вводить дополнительные характеристики или параметры обрабатываемых массивов. Первые пять символов блока ДБ содержат его шифр, а остальные 75 символов могут содержать произвольную информацию.

Программный обмен информацией массивов между оперативной и внешней памятью не предполагает однозначного соответствия между массивами и катушками магнитных лент. На одной катушке ленты могут быть размещены один или несколько массивов, или один массив может занимать несколько катушек ленты. В последнем случае необходимо вводить специальные промежуточные контрольные (начала и конца) части мас-

сива, расположенные на отдельной катушке магнитной ленты.

Параметры промежуточного начального контрольного блока (ПНБ) аналогичны параметрам блока НБ, за исключением символов 1—5-го (* * ПНБ) и 11—12-го, содержащих номер ПНБ массива, начиная с первого.

Параметры промежуточного конечного контрольного блока (ПКБ) совпадают с параметрами блока КБ, за исключением символов 1—5-го (* * ПКБ). Вслед за ПНБ и ПКБ не допускается использование блоков ДБ.

Варианты схем расположения массивов информации на катушках магнитной ленты показаны на рис. 3.20, 3.21, 3.22.

В качестве стандартного размера зоны в системе программирования принят размер в 1920 информационных символов. В пределах одного массива допускаются зоны разной длины от 5-го до 1920-го информационных символов. Нумерация зон массива сквозная, начиная с блока НБ, имеющего номер зоны, равный нулю. Следующая зона — номер 1 и т. д. до блока КБ или последнего блока ДБ, записанного вслед за КБ. Количество катушек магнитных лент, на которых хранится массив, для нумерации зон значения не имеет. Если на одной катушке ленты расположено несколько массивов, то нумерация зон начинается с блока НБ каждого из массивов.

Организация массивов информации на других носителях

Для массивов информации, подготовленных на перфоленге, используются контрольные блоки: НБ, КБ, ПНБ и ПКБ. Структура этих блоков аналогична ранее рассмотренным.

Количество символов контрольных блоков для перфоленга может быть меньше 80 и ограничивается символом *, перфорируемым после информации контрольного блока.

Возможные варианты схем расположения массива на перфоленгах показаны на рис. 3.23, 3.24.

Для массивов информации, подготовленных на перфокартах, используются только контрольные блоки НБ и КБ, которые занимают по одной перфокарте. Структура блоков соответствует ранее рассмотренным.

На перфокартах массивы получают либо в результате перфорации на УПДК, либо при выводе из машины. Количество информационных карт между блоками НБ и КБ не ограничено.

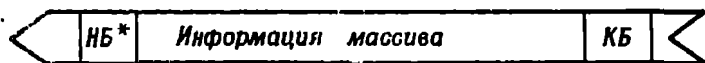


Рис. 3.23. Схема расположения информации одного массива на перфоленте

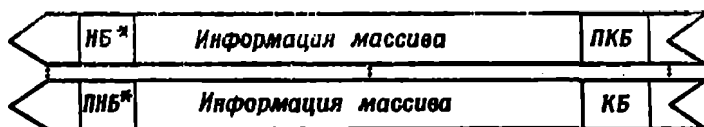


Рис. 3.24. Схема расположения информации одного массива на нескольких катушках перфоленты

3.3. ПРОЦЕДУРЫ РАБОТЫ С МАССИВАМИ

3.3.1. Сортировка информации

Под сортировкой принято понимать процедуру разделения записей на группы в соответствии с признаками, используемыми для их идентификации, или по определенным правилам. В зависимости от характера обработки информации в процессе сортировки принято выделять: распределение, упорядочение, подбор, выборку и объединение.

Сортировка по способу распределения предусматривает разделение сортируемых записей на группы, характеризующиеся одинаковым значением сортируемого признака. Примером такого вида сортировки может служить сортировка пачки перфокарт по какой-либо колонке. Для рассматриваемого вида сортировки характерно, что записи с одинаковым значением сортируемого признака собираются в одну группу, причем порядок расположения записей внутри группы не имеет значения.

Сортировка по способу упорядочения предусматривает расположение записей в порядке возрастания или убывания значений сортируемого признака. Примером такого вида сортировки является расположение записей в алфавитном порядке фамилий в списках, наимено-

ваний материалов в справочниках и т. д. Упорядочение имеет особенно важное значение в связи с тем, что это наиболее широко используемый прием сортировки записей.

Сортировка по способу подбора предусматривает нахождение для каждой записи из одной группы данных соответствующей записи с таким же значением сортируемого признака из другой группы данных. При этом записи обеих групп данных могут быть как упорядочены, так и не упорядочены. Примером подбора может служить подбор записей по одному разведываемому объекту из различных источников данных.

Сортировка по способу выборки предусматривает отбор из группы данных записей, характеризующихся определенными значениями признака. Например, выбор записей, принадлежащих одному и тому же объекту.

Сортировка по способу объединения предусматривает получение одной последовательности записей, построенной по определенному правилу, из двух и более последовательностей, построенных по тому же правилу. Операция объединения находит широкое применение в задачах учета и отчетности.

Сортировки информации способом подбора и выборки мало чем отличаются друг от друга, в них используются обычно общие приемы. Учитывая их значение для решения военных задач, рассмотрим реализацию этих методов.

Остановимся на сортировке распределением и упорядочением. Прежде всего необходимо отметить, что оба эти способа тесно связаны друг с другом. Это объясняется в первую очередь тем, что распределение может быть использовано в процессе упорядочения. Для этого необходимо только собрать в определенном порядке группы записей, полученных в результате распределения. Кроме того, упорядоченные записи являются в известном смысле и распределенными, так как для получения распределенных записей достаточно разделить упорядоченную последовательность записей на группы в соответствующих точках.

Основное отличие сортировки по способу распределения от сортировки упорядочением состоит в следующем:

— в каждой группе записей, полученной в результа-

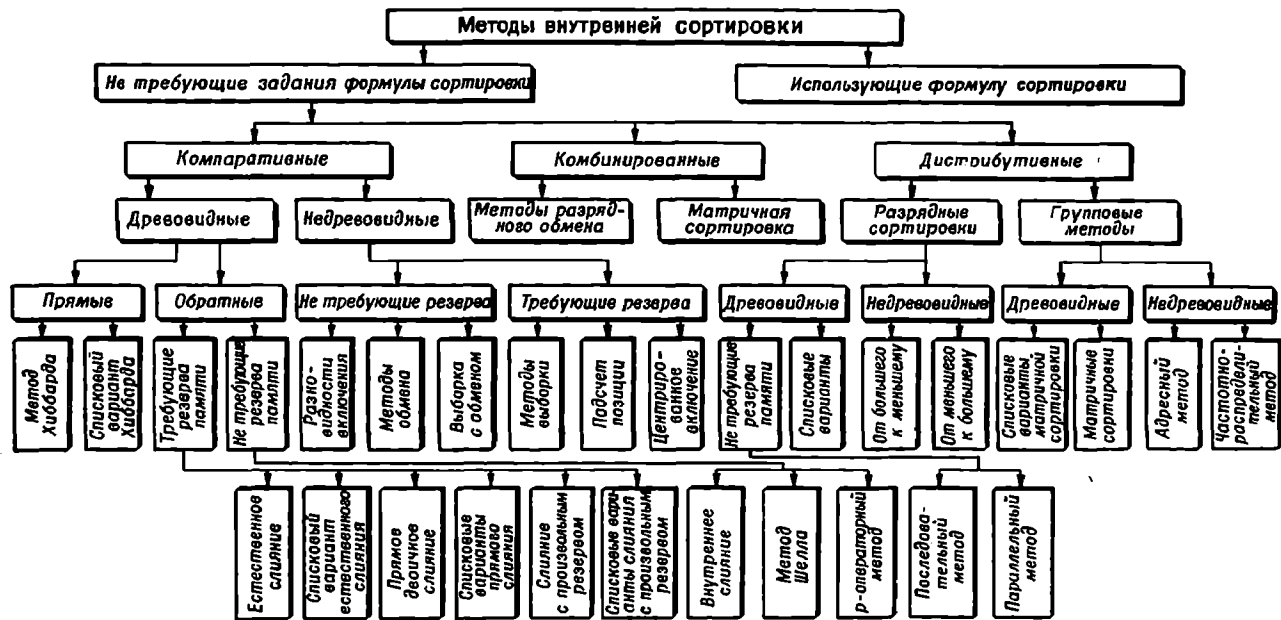


Рис. 3.25. Классификационная схема методов внутренней сортировки

те распределения, сохраняется тот относительный порядок, который имел место в исходных данных, а при упорядочении относительный порядок записей, как правило, нарушается;

— эффективность распределения существенно зависит от распределения признаков в исходной совокупности, а эффективность упорядочения от этого не зависит;

— в процессе упорядочения исходные и выходные данные обычно располагаются в одной и той же части оперативного накопителя, а при распределении — в различных частях.

При изучении сортировки информации с использованием ЭВМ принято выделять внутреннюю и внешнюю сортировку. Процесс сортировки, выполняемый над записями, размещенными в оперативной памяти ЭВМ, принято называть внутренней сортировкой. Практически объем сортируемых записей значительно превышает объем оперативной памяти. В этих случаях сортируемая информация записывается во внешней памяти и по мере обработки по частям вводится в оперативную память. Сортировку с применением внешней памяти принято называть внешней сортировкой.

Существующее многообразие методов внутренней сортировки информации (рис. 3.25) требует систематизации в целях определения их места и значения, выделения наиболее быстродействующих и последующей разработки программного обеспечения сортировки информации на ЭВМ.

Принято разделять методы внутренней сортировки на компаративные (сравнительные) и дистрибутивные (распределительные) в зависимости от того, каким образом выполняется процесс определения принадлежности записи к группе.

При компаративной сортировке признак рассматривается как единый объект, результат оценки которого непосредственно используется для включения сортируемой записи в некоторую последовательность промежуточных или окончательных записей.

При дистрибутивной сортировке признак анализируется по частям и сортировка выполняется в несколько этапов, на каждом из которых анализируется одинаковая часть всех признаков. Длина части определяется

размером резерва оперативной памяти, необходимого для дистрибутивной сортировки. Время выполнения дистрибутивной сортировки более существенно зависит от суммарной длины признака, чем время выполнения компаративной сортировки. При небольшой длине признака быстродействующие дистрибутивные методы обладают более высокой производительностью, чем наиболее производительные компаративные методы. При сложном признаке сортировки значительный выигрыш во времени дают компаративные методы сортировки.

Время сортировки компаративными методами определяется в основном числом сравнений признаков и количеством действий, необходимых для включения записей в последовательность рассортированных записей путем их перемещений, обменов или других действий. Наравне с указанными действиями на время реализации оказывают влияние вспомогательные операции, количество которых соответствует числу сравнений и действий по включению записей.

Широко применяемые компаративные методы, именуемые недревовидными, — различные методы обмена, включения, вставки, выборки, подсчета позиции и другие, — обладают следующими особенностями:

- шифры записей просматриваются последовательно;
- группы, к которым причисляются две сравнительные записи, выражают отношение между признаками сортировки этих записей;

- разделение исходной последовательности записей на несколько составных частей в целях получения упорядоченной последовательности не дает сокращения времени реализации сортировки;

- зависимость всех или некоторых компонентов операционного набора (число сравнений, обменов, пересылок и т. д.), определяющих производительность сортировки, от количества сортируемых записей имеет параболический характер.

Для наиболее производительных компаративных методов сортировки характерна тенденция к такому изменению алгоритма сортировки, которое приводит к взаимно сбалансированному увеличению связности между отдельными записями при поиске места включения в производную последовательность и относительному уменьшению связности между ее отдельными частями.

Основные отличия данных методов, называемых древовидными, от недревовидных компаративных методов состоят в следующем:

положение записей в формируемых последовательностях по отношению к положению в исходном массиве изменяется с неравномерным шагом, среднее значение которого убывает от этапа к этапу до минимального (в прямых древовидных методах) или возрастает до максимального (в обратных древовидных методах);

целью элементарных действий по упорядочению является относительно простая задача включения записи в одну из групп, составляющих промежуточную последовательность;

количество этапов, на которые организуется просмотр признаков всех записей, для древовидной сортировки меньше, чем для недревовидной;

для большинства древовидных методов зависимость числа сравнений, перемещений и обменов, приходящихся на одну запись, имеет логарифмический характер.

Одним из примеров древовидного метода сортировки является метод Шелла [2]. Идея метода состоит в том, чтобы в первую очередь обнаружить и упорядочить пары, состоящие из удаленных элементов, начиная с пар, в которых расстояние между записями составляет примерно $n/2$ позиций, затем просматривать пары с расстоянием $n/4$ и т. д. На заключительном этапе процесса упорядочиваются смежные пары, но за счет ранее выполненных перестановок объем работы на этом этапе существенно сокращается.

Рядом опытных работ установлено [10, 48], что если на очередном этапе группа формируется из записей различных групп, то быстродействие процесса возрастает. Этот процесс может быть описан в виде следующей процедуры:

```
procedure Shell sort ( $A, n$ ); value  $n$ ; integer  $n$ ;  
    integer array  $A$ ;  
    begin integer  $i, j, k, R$ ;  $k := n$ ;  
    for  $k :=$  if  $k > 2$  then  $(k - 1) \div 2$  else  $k \div 2$  while  $k \geq 1$  do  
        for  $i := k + 1$  step 1 until  $n$  do  
            begin if  $A[i - k] > A[j]$  then  
                begin  $R := A[i]$ ;  $j := i - k$ ;
```

```

      M : A [j + k]: = A [j];
      if j > k then
      begin if A [j - k] > R then
      begin j: = j - k; goto M end
      end; A [j]: = R
      end
      end
end procedure Shell sort;

```

Данный способ не требует для своей реализации дополнительной памяти и при кажущейся алгоритмической сложности достаточно просто программируется. Программная реализация метода Шелла приведена в работах [31, 48]. Этот способ реализован в программном обеспечении ЭЦВМ «Минск-32».

По сравнению с древовидными методами процесс сортировки в недревовидных компаративных методах характеризуется большим числом производимых действий. Для приближенной оценки методов можно воспользоваться данными табл. 5, приведенной в работе [20].

Практический интерес представляют списковые методы сортировки, при которых результатом анализа является преобразование вспомогательного адресного списка, содержащего n элементов, а расположение самих записей изменяется лишь один раз — при расстановке записей по своим местам. При наличии достаточного резерва памяти целесообразно использовать списковые алгоритмы слияния, отличающиеся высокой производительностью в широком диапазоне изменения параметров сортируемого массива (n — число записей, m — длина одной записи, N — совокупная длина признака сортировки).

К числу дистрибутивных методов сортировки, называемых чаще разрядными и представленных не столь широко, как компаративные, относятся матричная сортировка [6], адресный [40] и частотно-распределительный [10] методы. Метод разрядного обмена обладает некоторыми чертами, сходными с чертами древовидных компаративных методов, он также сходен с дистрибутивными методами, в которых признак анализируется тоже по частям.

Наименование метода			Среднее количество сравнений при одном сортируемом признаке (C)
Компаративные	Недревовидные	Обмен	$C \approx 0.5n^2 - 0.5n$
		Включение	$C \approx 0.25n^2$
		Выборка с обменом	$C \approx 0.5n^2$
		Подсчет позиции	$C \approx 0.5n^2$
		Центрированное включение	$C \approx 0.167n^2$
	Древовидные	Метод Хиббарда	$C \approx 1.4n \log_2 n$
		Метод слияния	$C \approx n \log_2 n$
		Метод Шелла	$C \approx \frac{n}{8} (\log_2 n)^2$
		P — операторный метод	$C \approx 3^r - 2^r$ при $n = 2^r$

Таблица 5

Среднее количество перемещений (P), обменов (Q) заключений в цепной список (I)	Удельный резерв памяти (машинных слов/запись)
$Q \approx 0.25mn^2$	0
$P \approx 0.25mn^2$	0
$P = (n-1)m; Q = (n-1)m$	0
$P = mn$	1
$P \approx 0.167mn^2$	m
$P = mn; I = n^*$	1
$Q \approx 0.35mn \log_2 n$	≈ 0
$Q \approx 0.35n \log_2 n; P = mn^*$	≈ 1
$P = mn \log_2 n$	m
$P = mn; I \approx n \log_2 n^*$	1
$Q \approx n/16m (\log_2 n)^2$	0
$Q = 3^r - 2^r$ при $n = 2^r$	0

Наименование метода		Среднее количество сравнений при одном сортируемом признаке (C)	Среднее количество перемещений (P), обменов (Q) включений в цепной список (I)	Удельный резерв памяти (машинных слов/запись)
Разрядные	Параллельный метод разрядного обмена	$C \approx Nn$	$Q \approx 0.5mn \log_2 n^{**}$	0
	Последовательный метод разрядного обмена	$C \approx n \log_2 n$	$Q \approx 0.25mn \log_2 n^{***}$	≈ 0
	Недревовидная разрядная сортировка	$C = Nn$	$P = \frac{n^2 Nm}{8}$	0
Комбиннированный метод		$C \approx n \log_2 n$	$Q \approx 0.25mn \log_2 n$	≈ 0

* Списковые варианты методов.

** При случайном расположении кодов признаков.

*** При случайном расположении и равномерном распределении по шкале признаков.

Каждый из методов, включенных в классификационную схему, показанную на рис. 3.25, отражен в отечественной литературе. Поэтому остановимся на тех требованиях, которые предъявляются к процедурам сортировки информации с точки зрения универсальности. Основные требования:

- реализация алгоритма для сортировки произвольного числа записей (в пределах оперативной памяти);

- возможность работы при различных положениях сортируемого массива в оперативной памяти;

- реализация сортировки произвольной длины записей, одинаковой для всего сортируемого массива;

- реализация алгоритма при любом размещении признака сортировки внутри записи;

- возможность использования в качестве признаков как слов с фиксированной, так и слов с плавающей запятой.

Естественно, что не каждый алгоритм удовлетворяет всем этим требованиям. Необходимость выбора алгоритма сортировки применительно к конкретным значениям параметров сортируемого массива диктуется тем обстоятельством, что методы сортировки имеют весьма различную производительность.

Сравнительная оценка некоторых методов внутренней сортировки информации (15 различных алгоритмов), выполненная на основании работы [48] с использованием специального испытателя программ по критериям минимума времени сортировки и занимаемого объема памяти, позволила исключить из рассмотрения по совокупному критерию большую часть алгоритмов. Остались для последующего анализа программы, разработанные для метода Шелла (СП-8), и сортировка обособленных признаков (СП-7).

На основании анализа функции эффективности СП-7 и СП-8 на рис. 3.26 показан график (кривая 1) равных эффективностей в координатах n (число сортируемых записей) и m (длина сортируемых записей), обеспечивающий быстрый выбор эффективного метода сортировки информации. В области графика над кривой 1 для внутренней сортировки информации целесообразно использовать алгоритм сортировки обособленных признаков, а в области под кривой 1 — алгоритм сортировки методом Шелла.

Кроме того, на график нанесены ограничительные кривые емкости оперативной памяти: кривая 2 — при использовании двух блоков памяти (8192 слова для ЭЦВМ «Минск-22») — и кривая 3 — при использовании одного блока памяти (4096 слов).

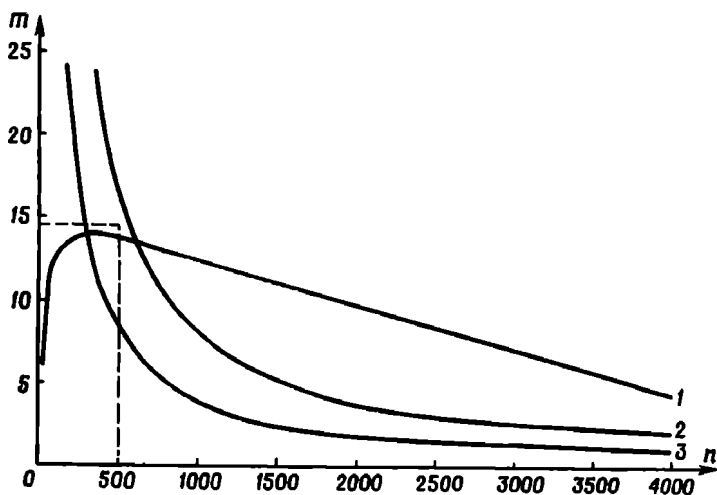


Рис. 3.26. График сравнительной характеристики методов внутренней сортировки

Для выбора оптимального алгоритма внутренней сортировки информации с использованием графика необходимо знать параметры сортируемого массива (n и m) и ограничение по памяти.

При внешней сортировке исходные данные размещаются во внешней памяти. Действия по сортировке выполняются в несколько этапов, и внешняя память при этом неоднократно используется для размещения данных. Для ускорения процесса сортировки данных либо на конечных, либо на начальных этапах внешнего упорядочения используется внутренняя сортировка части массива данных, соизмеримого с объемом оперативной памяти, выделенной для работы с внешней памятью.

При использовании внешней памяти последовательного типа возникает ряд специфических различий между внешней и внутренней сортировкой. Дело в том, что в силу сравнительно низкой скорости обмена информа-

цией между оперативной и внешней памятью на оценке производительности метода в значительной мере сказывается организация обмена или, другими словами, фаза внутреннего упорядочения при внешней сортировке не играет существенной роли. Классификационная схема методов внешней сортировки, показанная на рис. 3.27, учитывает эту особенность [38].

Разделение методов внешней сортировки на основные разновидности (прямые и обратные древовидные, недревовидные разрядные) дается в указанной работе в общем плане, так как отражает основные различия в построении методов и их оценках. В качестве критерия второго яруса классификационной схемы рассматривается способ разделения на группы при обмене с внешней памятью.

В двоичных методах разделение внешнего упорядоченного сегмента производится посредством сравнения признаков записей с некоторым «опорным» признаком. Полученные подгруппы (текущий признак меньше «опорного» и больше «опорного») образуют новые сегменты.

При K -ичных методах для разделения сегмента используется $(K-1)$ «опорный» признак. При случайном характере выбора признаков сегмент делится на неравные части, поэтому среднее число просмотров и пересылок каждой записи при выводе на ленту больше, чем при слиянии по K -путям.

В целях повышения производительности методов необходимо осуществлять предварительную оценку «опорного» признака в ходе предыдущего этапа упорядочения.

Очередное разделение на группы одного из только что полученных сегментов называется последовательным методом. Процесс реализации прямых древовидных методов можно изобразить в виде дерева, в котором каждый узел отображает внешне упорядоченный сегмент, ветви, исходящие из узла, соответствуют разделению сегмента, а корень дерева отображает исходную последовательность записей.

Последовательное разделение сегментов, изображаемых узлами одного яруса, называется параллельным методом.

Процесс реализации последовательного метода позволяет частично совместить во времени подвод очеред-

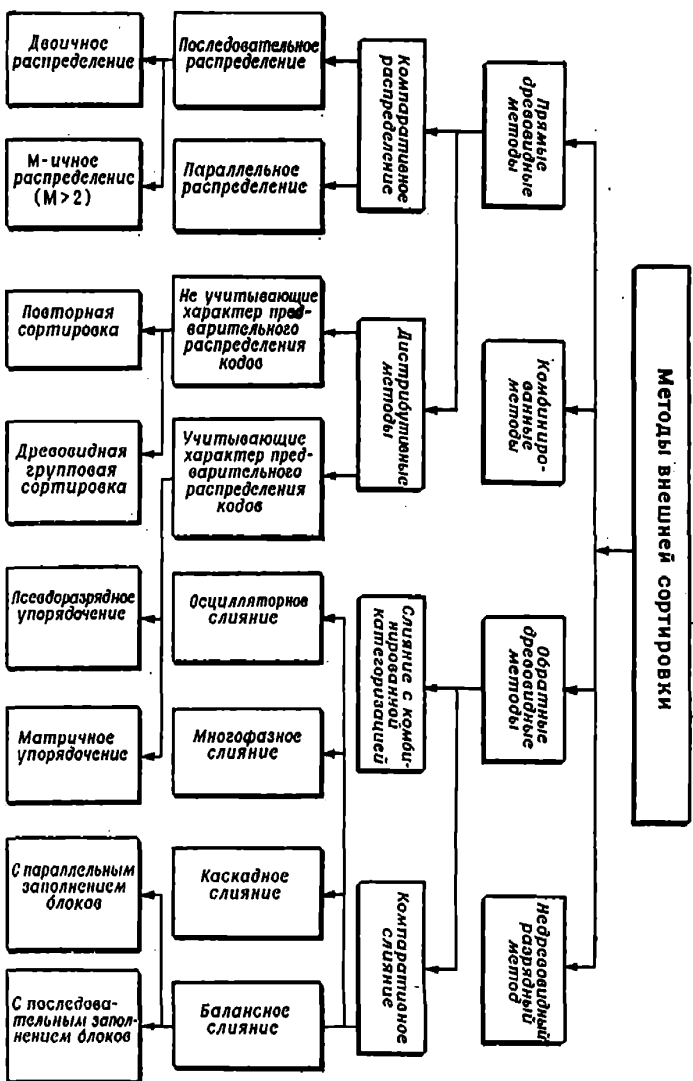


Рис. 3.27. Классификационная схема методов внешней сортировки

ных участков магнитной ленты с внутренним упорядочением предыдущих сегментов. Этот метод в отличие от параллельного позволяет получить нужную производительность при использовании минимального числа вспомогательных лент [34].

Распределительные методы сортировки с разделением на группы с переписью на НМЛ достаточно подробно изложены в литературе и подразделяются на повторную сортировку, древовидную групповую сортировку, псевдоразрядное упорядочение [34] и матричное упорядочение [28]. Их отличительной особенностью является практическая независимость от предварительного упорядочения массива.

Обратные древовидные методы сортировки представлены в классификационной схеме разновидностями слияния. Их отличительными особенностями являются:

- независимость времени упорядочения от характера распределения признаков по числовой оси и их расположения в массиве;

- возможность заранее регламентировать величины предельных сегментов, получающихся в ходе упорядочения.

Анализ разновидностей слияния [20] показывает, что каждая из них наиболее эффективна для определенной совокупности данных, общий разброс приближенных оценок их производительности не превышает 30%.

Комбинированные методы сочетают в себе достоинства прямых и обратных древовидных методов.

При недревовидном разрядном упорядочении сегменты последовательности данных, передаваемые между внутренней и внешней памятью, не связаны с внутренней сортировкой последовательности, а все этапы разделения записей на группы, число которых определяется разрядностью признака, являются этапами переписи данных во внешнюю память.

При выборе более производительного метода внешней сортировки больших массивов необходимо учитывать, что основное время приходится на подвод заданных зон носителя информации и обмен между внешней и оперативной памятью. При организации алгоритма внешней сортировки следует учитывать наличие вспомогательных внешних носителей, совмещение подвода с выполнением других операций, очередность использова-

ния внешних носителей, возможность их параллельной работы.

На практике наиболее часто применяется метод, в котором используется слияние частей массива, расположенных на различных лентах, например, метод внутреннего объединения, модифицированный для внешней сортировки (метод фон Неймана).

Сущность метода состоит в следующем. Упорядоченность массива наращивается последовательно по этапам. На первом этапе добиваются того, чтобы массив распался на упорядоченные части, включающие конечное (сравнительно небольшое) число записей. На втором этапе производится объединение и внутреннее упорядочение двух частей, а после K -го этапа длина упорядоченной последовательности частей должна быть не менее двух. Тогда весь массив окажется упорядоченным максимум через $\text{entier}(\log_2 n) + 1$ этапов. Если на каждом этапе число сравнений признаков и перестановок записей будет равно n , то упорядочение закончится не более чем через $n \log_2 n$ этапов.

На каждом этапе метода используются две входные и две выходные ленты. При этом входные ленты содержат одинаковое (расхождение не более одного) число участков сортируемого массива, каждый из которых внутренне упорядочен.

Алгоритмом сортировки предусматривается предварительный этап, в результате которого исходный массив последовательно порциями считывается в оперативную память, упорядочивается одним из внутренних алгоритмов сортировки и выводится поочередно (по порциям) на входные ленты.

Оперативная память делится на три участка (сегмента); два из них загружаются порциями (по одной) с входных лент. Записи, находящиеся во входных сегментах, объединяются в выходной сегмент, который по мере заполнения выгружается на одну из выходных лент, затем продолжается объединение записей входных сегментов.

Следящая система позволяет устанавливать момент завершения обработки записей каждого из входных сегментов с их последующей загрузкой с соответствующих входных лент. Эти действия выполняются до тех пор, пока на одной из входных лент не завершится выборка

участка. В этой ситуации содержимое второго (незавершенного) участка переписывается на выходную ленту.

По завершении объединения очередных участков сегмент выходной ленты настраивается на запись на вторую выходную ленту, а во входные сегменты производится загрузка очередных порций входных лент. Непарный участок на одной из входных лент (если он есть) переписывается через оперативную память на ту из выходных лент, которая должна была принять очередные два участка, если бы существовала пара.

Очередной этап заканчивается после просмотра всех участков входных лент и формирования выходных лент. На следующем этапе входные и выходные ленты меняются ролями, устанавливаются в исходное состояние и начинается новый этап.

На последнем этапе входные ленты содержат лишь по одному участку, в результате слияния которых на одной из выходных лент получается упорядоченный массив.

Приведем запись данного алгоритма внешней сортировки информации в виде процедуры на АЛГОЛ-60 при следующих допущениях. Исходный массив размещен на магнитной ленте j в виде N порций по l целых чисел в каждой. Кроме этой ленты в процессе работы участвуют ленты $j1$, t и $t1$. Параметры процедуры N , l , j , $j1$, t , $t1$ — целые числа. Параметр t вызывается по наименованию, так как по окончании работы процедуры значением соответствующего фактического параметра должен быть номер магнитной ленты, где записан отсортированный массив. Работа с магнитными лентами выполняется процедурами:

$\text{read } (j, b, i, l)$, осуществляющей считывание информации с магнитной ленты в ячейки оперативной памяти: $b[i]$, $b[i+1]$, ..., $b[i+l-1]$ — очередную зону из l чисел с ленты номер j ;

$\text{write } (j, b, i, l)$, осуществляющей запись на магнитную ленту;

$\text{rewind } (j)$, осуществляющей подвод ленты в исходное положение;

$\text{sort } (b, i, l)$, осуществляющей внутреннее упорядочение.

Алгоритм процедуры слияния:

```

procedure tape Neumann sort ( $N, l, j, j1, t, t1$ );
  value  $N, l, j, j1, t1$ ; integer  $N, l, j, j1, t, t1$ ;
begin integer  $i, i1, i2, k, k1, m, r, r1, s, nj, nj1, nt, nt1$ ;
  integer array  $b[1 : 3 \times l]$ ; switch  $Q := M1, M2, M3$ ;
  procedure advance ( $i, k, j, r, a, s$ , switch, empty);
    value  $j, r, s$ , switch; integer,  $i, k, j, r, s$ , switch;
    integer array  $a$ ; label empty;
    if  $i > l$  then
      begin if  $k < r$  then
        begin read ( $j, a, s, l$ );  $k := k + 1$ ;  $i := 1$  end
        else begin  $m := \text{switch}$ ; goto empty end
        end advance;
      procedure rotate ( $i, j$ ); integer  $i, j$ ;
      begin integer  $k$ ;  $k := i$ ;  $i := j$ ;  $j := k$ ; end rotate;
      rewind ( $j$ ); rewind ( $t$ ); rewind ( $t1$ );  $nt := nt1 := 0$ ;
      for  $s := 1$  step 1 until  $N$  do
        begin rotate ( $t, t1$ ); rotate ( $nt, nt1$ );
        read ( $j, b, 1, l$ ); sort ( $b, 1, l$ );
        write ( $t, b, 1, l$ );  $nt := nt + 1$ 
        end;  $nj := nj1 := 0$ ;  $s := 1$ ;
         $L : \text{if } nt < N \text{ then}$ 
          begin rewind ( $j$ ); rewind ( $j1$ ); rewind ( $t$ );
          rewind ( $t1$ ); rotate ( $j, t$ ); rotate ( $j1, t1$ );
          rotate ( $nj, nt$ ); rotate ( $nj1, nt1$ );
           $L1 : i := i1 := l + 1$ ;  $i2 := 1$ ;  $k := k1 := 0$ ;  $m := 1$ ;
           $r := \text{if } nj \geq s \text{ then } s \text{ else } nj$ ;
           $r1 := \text{if } nj1 \geq s \text{ then } s \text{ else } nj1$ ;
           $M1 : \text{advance } (i, k, j, r, b, 1, 3, M3)$ ;
          advance ( $i1, k1, j1, r1, b, b + 1, 2, M2$ );
          if  $b[i] < b[l + i1]$  then
             $L2 : \text{begin } b[2 \times l + i2] := b[i]$ ;  $i := i + 1$  end
            else
               $L3 : \text{begin } b[2 \times l + i2] := b[l + i1]$ ;  $i1 := i1 + 1$  end;
               $i2 := i2 + 1$ ;
              if  $i2 > l$  then

```

```

begin write ( $t, b, 2 \times l + 1, l$ );  $nt := nt + 1$ ;  $i2 := 1$  end
      goto Q [ $m$ ];
M2: advance ( $i, k, j, r, b, 1, 1, M4$ ); goto L2;
M3: advance ( $i1, k1, j1, r1, b, b + 1, 1, M4$ ); goto L3;
      M4:  $nj := nj - r$ ;  $nj1 := nj1 - r1$ ;
if  $nj > 0 \vee nj1 > 0$  then begin rotate ( $t, t1$ ); rotate ( $nt, nt1$ );
      goto L1 end;  $s := 2 \times s$ ; goto L end end;

```

В состав программного обеспечения ЭЦВМ «Минск-32» включена библиотечная программа внешней сортировки массива записей постоянной длины [31], размещенного на магнитной ленте. В этой программе использован метод двухступенчатого балансного объединения. Упорядочение производится следующим образом.

С исходной магнитной ленты последовательно считываются порции информации в отведенное поле оперативной памяти. Считанная информация упорядочивается с помощью одной из программ внутренней сортировки. Упорядоченные порции записываются поочередно на одну из двух рабочих магнитных лент (МЛ1 и МЛ2). Объединение упорядоченных последовательностей, размещенных на МЛ1 и МЛ2, производится так: область оперативной памяти для внутренней сортировки делится на три равные подобласти. Две из них используются в качестве областей ввода, третья — в качестве области вывода.

С каждой из двух рабочих лент (МЛ1 и МЛ2) считывается по одной порции в области ввода, производится их объединение и попеременная запись на рабочие ленты МЛ3 и МЛ4. Далее смена лент производится так же, как и по способу Неймана. Этот процесс продолжается до тех пор, пока на одной из рабочих магнитных лент не получится упорядоченная последовательность, которую можно переписать на личную ленту пользователя.

3.3.2. Выборка информации

Одной из типичных процедур обработки больших массивов является выборка информации. Сущность выборки состоит в поиске и отборе из организованных или неорганизованных массивов записей, обладающих определенными значениями признака.

Для реализации выборки необходимо задать объем массива, из которого выполняется выборка, его местоположение, длину записи, местоположение признака отбора в записи, признак выборки и местоположение массива результатов выборки. Для случая групповой выборки необходимо задавать пределы признаков выборки.

При отборе по части признака указывается местоположение и количество разрядов признака, по которым осуществляется выборка.

Алгоритмом выборки (рис. 3.28) предусматривается последовательный просмотр записей в целях определения принадлежности к заданной группе (по признаку). Записи, признаки которых совпадают по значению с

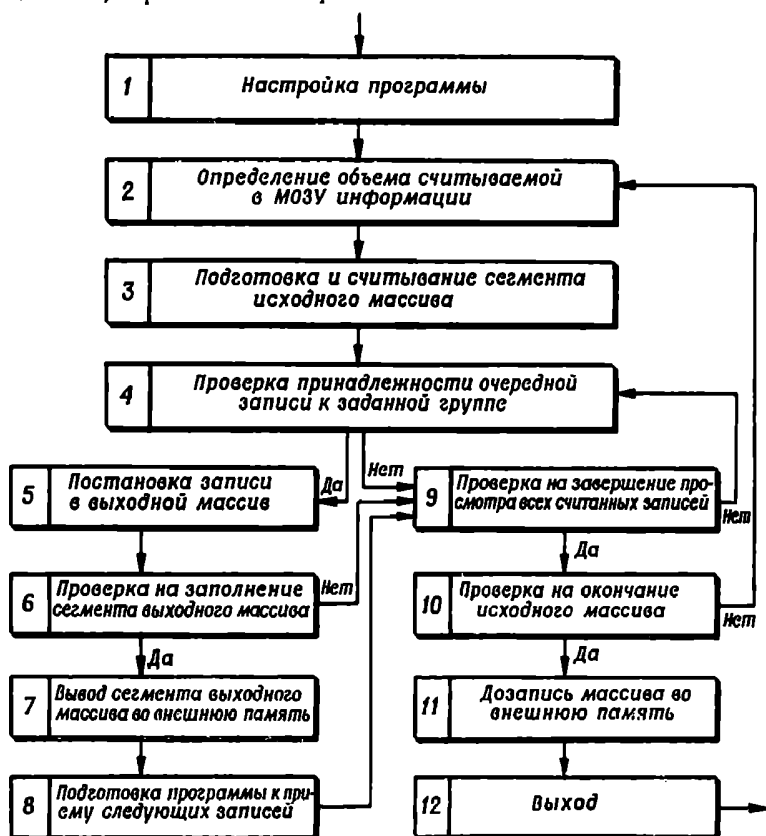


Рис. 3.28. Блок-схема алгоритма внешней выборки

признаком выборки, размещаются в выходном массиве; выполняется их подсчет.

Настройка на параметры выборки выполняется программой блока 1. Специфика обработки больших массивов информации при ограниченной оперативной памяти требует реализации автоматической настройки на оптимальный вариант распределения оперативной памяти под массивы в целях минимизации общего количества обращений к внешней памяти. Эти действия реализуются программой блока 2.

Программа блока 3 производит считывание части массива из внешней памяти с свободным доступом и определение количества считанных записей.

Проверка принадлежности каждой из считанных записей к заданной группе (признаку) осуществляется программой блока 4. Запись, удовлетворяющая заданным условиям выборки, размещается в сегменте выходного массива (блок 5) с проверкой на его заполнение. По мере заполнения сегмента выходного массива предусматривается выгрузка выбранных записей (блок 7) во внешнюю память с переадресацией местоположения свободной памяти и установкой фиксатора свободного сегмента в нулевое положение (блок 8).

После завершения последовательного просмотра записей сегмента, считанного в оперативную память (блок 9), предусмотрена проверка на завершение просмотра всех сегментов исходного массива программой блока 10.

Для принятого механизма вывода выбранных записей во внешнюю память по мере заполнения сегмента требуется дополнительная проверка на необходимость дозаписи выбранной информации, если сегмент не заполнен. Эти действия реализуются программой блока 11.

Настройка программы на выборку по заданному признаку, их диапазону или группе значений реализуется инвариантной программой блока 4.

После завершения работы программы количество выбранных записей передается стандартным образом для других программ или (и) выводится на печать.

Алгоритм выборки для случая организованного массива исходной информации значительно упрощается за счет целенаправленного просмотра информации.



Рис. 3.29. Классификационная схема способов поиска информации

3.3.3. Поиск информации

Поиск информации в массивах данных является для некоторых задач одной из основных процедур обработки информации. В этих случаях на поиск информации приходится большая часть машинного времени.

Поиск при существующих типах внешней памяти осуществляется в два этапа. На первом этапе определяется адрес сопровождающей информации, а на втором этапе по найденному адресу осуществляются адресация к внешней памяти и выборка заданной единицы информации в оперативную память. Под сопровождающей информацией понимается информация, характеризующая запись или отдельный ее элемент с точки зрения принадлежности к определенному объекту.

Способы организации хранения и поиска информации тесно связаны друг с другом. Более того, эффективность поиска в значительной мере зависит от способа организации хранения информации.

В основу классификации по способам поиска могут быть положены следующие принципы:

1) принцип отображения множества исходных слов на множество адресов:

- нахождение адреса слова (признака) постановкой его в соответствие совокупности признаков, хранящихся во внешней памяти;

- преобразование кода признака в адрес записи (характеристики);

2) по характеру поискового образа признака:

- полный код исходного признака;

- часть кода исходного признака;

- преобразованный код исходного признака.

Применительно к перечисленным выше принципам классификации на рис. 3.29 приводится классификационная схема возможных способов поиска информации.

Рассмотрим разновидности способов поиска и алгоритмы их реализации, исходя из предположения, что имеется массив из N записей, признаки которых размещены каждый в одной ячейке, и оценим среднее время поиска.

Способ перебора

Разновидности алгоритмов способов перебора (сканирования) предусматривают последовательный просмотр записей массива и сравнение их признаков с заданным признаком поиска.

При поиске способом перебора возможна одна из следующих разновидностей организации хранимого массива:

1) записи упорядочены в порядке возрастания (убывания) признака;

2) записи упорядочены в порядке убывания частоты использования признаков;

3) записи хранятся в массиве неупорядоченно, по мере их поступления.

Минимально возможное время поиска записи с заданным признаком определяется временем выполнения одного цикла поиска, включающего совокупность арифметических (логических) операций и операций управления, обеспечивающих идентификацию признаков.

Максимальное время поиска по способу перебора определяется временем просмотра всех признаков записей

$$T_{\max} = N.$$

Среднее время поиска одной записи определяется как математическое ожидание

$$T_{\text{ср}} = M(x) t_i = \sum_{i=1}^N p_i t_i, \quad (1)$$

где p_i — вероятность появления i -го признака в массиве записей;

t_i — продолжительность поиска i -го признака (в циклах поиска).

Если все признаки равновероятны, т. е.

$$p_1 = p_2 = \dots = p_N \approx \frac{1}{N},$$

то, считая $t_i = i$ циклам поиска, получим, что для определения местоположения записи в среднем требуется

$$T_{\text{ср}} = \frac{N+1}{2} \approx \frac{N}{2}$$

циклов поиска, так как для больших массивов $N \gg 1$.

При хранении записей в порядке убывания частоты использования оно может быть снижено до величины

$$T_{\text{ср}} = \frac{N}{\log N} \quad (2)$$

циклов поиска.

Отметим, что определение частоты использования записей является достаточно трудоемкой работой, базирующейся на статистическом накоплении и последующей обработке экспериментальных данных.

Способ перебора имеет простой алгоритм реализации, но при большом N применение его нецелесообразно из-за больших временных затрат.

Поиск делением на части

При поиске делением на части уменьшается время поиска за счет ускоренной локализации уменьшенного объема массива записей, включающих искомую запись, с последующим поиском сканированием.

Способы поиска делением на части разделяются в зависимости от значения делителя и различаются по эффективности алгоритмов реализации поиска и сложности.

При способе, базирующемся на делении массива записей пополам, предполагается, что все записи в массиве упорядочены в порядке возрастания (убывания) значений признаков. Сначала признак поиска сравнивается с признаком средней записи массива. При совпадении признаков поиск завершен. Если код заданного признака больше кода признака текущей записи, то поиск необходимо продолжить в последовательности записей

$$\left[\frac{N}{2} + 1, N \right].$$

В противном случае поиск продолжается в последовательности записей

$$\left[1, \frac{N}{2} - 1 \right].$$

Процесс деления пополам продолжается до получения последовательности из двух или трех записей, среди которых осуществляется поиск сканированием.

Максимальное время поиска делением пополам выражается формулой

$$T_{\max} = E \{\log_2 N\} + 1 \text{ циклов,} \quad (3)$$

где E — значение целой части выражения в фигурных скобках.

Среднее время поиска может быть оценено по формуле (1).

В процедуре *SEARCH* (приводится ниже), реализующей алгоритм поиска делением пополам, параметр A есть массив признаков с границами $[1:n]$, AO — заданное значение признака, i — номер элемента, имеющего значение признака AO (результат работы процедуры), и P — количество элементов группы, в которой производится последовательный поиск.

Если в результате последовательного поиска среди P элементов не найдено значение, соответствующее AO , то производится контрольный останов с индикацией признака AO .

Предполагается, что признак состоит из одного поля, принимающего целочисленное значение, массив A упорядочен в порядке возрастания значений признака.

```

procedure search ( $A, n, P, i, AO$ );
value  $n, P$ ; integer  $n, P, i, AO$ ; integer array  $A$ ;
    begin integer  $x, y$ ;
     $x := 1$ ;  $M1$  : if  $n \leq P$  then goto  $M3$ ;
                 $y := n \div 2$ ;
    if  $AO < A[x + y]$  then goto  $M2$ ;
     $x := x + y$ ;  $n := n - y$ ; goto  $M1$ ;
     $M2$  :  $n := y$ ; goto  $M1$ ;  $M3$  :  $y := x + y - 1$ ;
    for  $i := x$  step 1 until  $y$  do
    if  $AO = A[i]$  then goto  $M4$ ;
    stop ( $AO$ );  $M4$  : end;
  
```

Способ поиска делением пополам можно улучшить (повысить эффективность поиска), если организовать хранение записей в массиве в порядке уменьшения частоты использования записей при поиске. При этом поиск выполняется одним из способов в «верхней» части массива, а если искомая запись в нем не обнаружена, то организуется поиск в «нижней» части массива.

Последовательное деление массива записей может быть выполнено в общем случае на K целочисленных частей, каждая из которых включает искомую запись и снова подвергается делению на K целочисленных частей. Процесс локализации искомой записи производится до некоторого фиксированного значения части массива, в которой осуществляется поиск сканированием.

В работах некоторых авторов показано, что оптимальное значение $T_{\max}$ при комбинированном методе поиска достигается при $K=2$. Показывается, что для $K=2 \div 7$ значение $T_{\text{ср}}$ получается практически одинаковым при повышении сложности программ обработки с ростом K .

Среднее время поиска адреса одной записи зависит от количества записей, величины шага деления K и может быть определено из условия

$$\frac{K}{2} E \{\log_2 N\} + 1 \leq T_{\text{ср}} \leq \frac{K}{2} E \{\log_2 N\} + (K-1) \quad (4)$$

или

$$T_{\text{ср}} \approx \frac{K}{2} (E \{\log_K N\} + 1) \quad (5)$$

с погрешностью не более $\frac{K-1}{2}$ при $K \geq 2$.

Максимальное время поиска можно оценить по формуле

$$T_{\max} \approx (K-1) \left(E \{\log_K N\} + \frac{1}{2} \right) \quad (6)$$

с погрешностью не более $\frac{K-1}{2}$.

Завершение поиска после локализации некоторой группы записей, включающих искомую, осуществляется сканированием. Минимальная величина (l) этой группы записей может быть определена из условия

$$l > 1 - \frac{\ln(K-1)}{\ln K}.$$

Ступенчатый поиск [12] целесообразно применять тогда, когда массив записей хранится в группах различной длины при наличии ссылок на место хранения групп во внешней памяти. На первом этапе ступенчатого поиска локализуется группа, в которой хранится искомая за-

пись, а на втором этапе (при отсутствии более глубокой градации по ступеням) осуществляется поиск сканированием.

Среднее время поиска записи может быть получено как сумма математических ожиданий среднего времени поиска сканированием на каждой ступени, включая и поиск сканированием внутри искомой ступени:

$$T_{\text{ср}} = M_1(x_i)t_1 + M_2(y_j)t_2 + \dots + M_{\varphi-1}(z_{\xi})t_{\varphi-1} + \\ + \frac{N_{\varphi}}{2}t_{\varphi} = t_1 \sum_1^{C_1} ip_i + t_2 \sum_1^{C_2} jp_j + \dots + t_{\varphi-1} \sum_1^{\varphi-1} \xi p_{\xi} + \\ + t_{\varphi} \frac{N_{\varphi}}{2}, \quad (7)$$

где $M_{\varphi-1}(z_{\xi})t_{\varphi-1}$ — математическое ожидание среднего поиска на предпоследней ступени;

p_{ξ} — вероятность появления ξ -го признака ступени;

$t_{\varphi-1}$ — среднее время поиска ξ -го признака ступени;

N_{φ} — количество признаков, которые просматриваются сканированием на последней ступени;

t_{φ} — среднее время поиска заданной записи сканированием;

$i = 1, 2, \dots, C_1; j = 1, 2, \dots, C_2; \dots; \xi = 1, 2, \dots, N_{\varphi}$.

При записи этой формулы предполагалось, что количество признаков на одной ступени не зависит от другой ступени. Программа реализации этого способа значительно сложнее программ других способов деления на части.

Способ переменного соответствия

Алгоритм поиска по данному способу [35] базируется на специальном представлении кодов признаков и обеспечивает ассоциативный поиск записи при адресации одновременно ко всем адресам ЗУ.

Время поиска одного многозначного признака выражается формулой

$$T_{\text{п}} = 2 \log_2 N. \quad (8)$$

Данный способ адресации требует удвоенного объема специальной памяти, что является существенным недостатком.

Способы поиска по дереву

Данным способом выполняют поиск в массивах с иерархической структурой, организованных для хранения в памяти по графу типа дерева. Существующие способы организации деревьев определяют алгоритмы поиска в них. Так, например, для поиска внутри массива, организация которого описывается двоичным деревом, можно использовать ссылки на левую и правую вершины, входящие в последовательность, т. е. двоичное дерево можно представить двухсвязным списком.

Поиск заданного признака начинается с вершины и производится путем сравнения этого признака с признаком текущей вершины. Если признаки совпадают, то заданная запись найдена. Если заданный признак меньше признака текущей вершины, то, используя левую ссылку, выполняется переход к очередной вершине. В противном случае осуществляется переход по ссылке к правой вершине. Процесс ветвления продолжается до тех пор, пока не будет найдено заданное значение признака, если оно имеется в массиве, либо выполнен выход на конец ветвления.

Количество циклов поиска, необходимых для обнаружения заданной записи, определяется длиной пути в двоичном дереве до необходимой точки иерархии. В каждой точке этого пути сравниваются признаки и при необходимости организуется ветвление. Среднее время поиска записи в этом случае определяется как

$$T_{\text{ср}} = 2 \left(\frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{N+1} \right) \approx 1,4 \log_2 N. \quad (9)$$

Если последовательность вершин упорядочить по возрастанию абсолютных значений и принять за начало поиска вершину в середине последовательности, то можно получить среднее время поиска, как и для случая деления пополам. В связи с тем что при создании дерева строятся ссылки, то в отличие от способа деления последовательности пополам при данном способе вычислять адрес очередной вершины не требуется. В случае по-

строения дерева из записей, обладающих разноразностным характером использования, можно выполнять поиск с меньшей величиной $T_{\text{ср}}$.

Одним из способов двоичного дерева является экономичный способ размещения информации, для которого вводятся ссылки на следующую вершину списка и на место хранения записи данной вершины. Этот способ называют двухадресным списковым способом [13]. Поиск заданной записи выполняется последовательным просмотром признаков вершин.

Для других списковых способов организации хранения информации (гнездовой и узловой списки, л-ичное дерево) алгоритмы поиска полностью определяются механизмом организации списков.

Способы вычисления адреса

В идеальном случае можно осуществить поиск в массиве любого размера за один шаг, если по значению признака представится возможность непосредственно определить (вычислить) адрес записи в памяти. Такой замысел можно реализовать при соответствующей организации хранения информации путем преобразования признака в адрес или вычисления адреса.

Преобразование признака в адрес можно выполнить как по части, так и по полному значению признака. Одним из способов преобразования части признака в адрес является способ открытой адресации [17], сущность которого состоит в следующем. Имеется N признаков, каждый из которых включает фиксированное количество десятичных цифр. В качестве адреса размещения признака в массиве берутся некоторые три десятичных разряда этого признака. В целях исключения неоднозначности преобразования признаков множества N в адреса памяти последнюю разделяют на секции. Адрес секции определяется одним из возможных чисел в интервале от 1 до A^i , где $A^i = 10^3$ для случая, если номер включает три десятичных разряда. Средняя длина секции составляет

$$L_0 = \frac{N}{A^i}. \quad (10)$$

В связи с тем что отображение множества N на множество A^i отличается от равномерного, возможно появ-

ление такого количества одинаковых кодов-адресов, которое переполнит установленную длину секции.

При образовании массива адрес хранения признака вырабатывается по его части в интервале $(1, A^i - 1)$. Это практически адрес секций, где должен храниться данный признак. Для записи признака проверяют наличие места в i -й секции, и если в ней есть место, то производится запись. Если i -я секция заполнена полностью, то обращаются последовательно к $i + 1$, $i + 2$ и т. д. секциям до тех пор, пока не будет найдено место для размещения признака.

При организации поиска в этом случае поступают аналогично. Например, если задан адрес 997, то поиск начинается с 997-й секции и продолжается при необходимости в 998, 999, 000, 001-й и т. д. до тех пор, пока не будет найден заданный признак или не будет завершен просмотр всей последовательности.

Одна из модификаций открытого способа адресации, обеспечивающая уменьшение времени поиска, состоит в следующем. Очередной признак располагается как можно ближе к своему вычисленному адресу. Для реализации размещения и последующего поиска признака выполняется проверка занятости вычисленного адреса. Если вычисленный адрес занят, то организуется просмотр адресов в порядке возрастания до тех пор, пока не будет найдена ячейка, содержащая признак такого же или меньшего вычисленного адреса. Последовательность признаков, хранящаяся в непросмотренной части секции, сдвигается, освобождая место для размещения очередного признака.

Таким образом, вычисленный адрес будет всегда меньше или равен фактическому адресу размещения признака. Известны и другие модификации этого способа [17].

Комбинированный способ отличается от ранее изложенных тем, что в качестве адреса здесь используется часть сжатого кода признака. Адрес может быть составным, включающим номер лентопротяжного механизма, номер зоны и номер слова в зоне [17].

Другим возможным способом преобразования признака в адрес является использование для преобразования полного кода признака. В основу всех способов положен принцип сжатия кода. Одним из известных спосо-

бов преобразования является способ цепной связи [17], сущность которого состоит в следующем. В результате преобразования признака вырабатывается адрес, но все признаки с одинаковым вычисленным адресом связаны между собой цепочкой переадресации. Предполагается, что при данном способе преобразование признаков в их адреса осуществляется в соответствии с законом распределения случайных чисел при равновероятном обращении к любой из записей массива.

Использование сжатых кодов признаков в качестве адресов этих признаков (записей) позволяет осуществить способ вероятностной адресации. В основе этого способа — зависимость образования сжатых кодов от вероятности появления символов алфавита в признаке и семантических особенностей языка.

Код преобразованного признака, получаемый на выходе блока сжатия или вырабатываемый программным способом, состоит из двух частей:

n_i — сжатого кода признака, т. е. его адреса;

t_i — кода сопровождающей информации.

Полный код преобразованной информации ($n_i + t_i$) однозначно характеризует любую запись во всем их множестве. При поиске организуется выборка по адресу n_i . Далее считывается и сравнивается сопровождающая информация. Если $t_{i, \text{всх}} = t_{i, \text{зу}}$, то запись выдается в оперативную память. Если $t_{i, \text{всх}} \neq t_{i, \text{зу}}$, то осуществляется переадресация на следующую запись внутри группы с адресом n_i .

Среднее время поиска записи оценивается для случая вероятностной адресации величиной [17]:

$$T_{\text{ср}} = \frac{\sum_{i=1}^{\eta_k} \frac{\eta(\eta+1)}{2\eta!}}{\sum_{i=1}^{\eta_k} \frac{\eta}{\eta!}}, \quad (11)$$

где η — тип группы неоднозначности сжатия, который обозначает, сколько признаков, имеющих одинаковый сжатый код, находятся в одной (η_i -й) группе при $i=1, 2, \dots, k$.

Ограничиваясь набором величин $\eta=1, 2, \dots, 10$, получим

для случая поиска одной записи при вероятностной адресации

$$T_{\text{ср}} = 1,5 \text{ цикла.} \quad (12)$$

Расчетные способы вычисления адреса записи представлены в классификационной таблице интерполяционным способом, способом деления полинома на полином и расчетом нумерации сочетаний.

Если имеется приемлемый алгоритм непосредственного вычисления адреса по признаку записи, то отпадает необходимость хранения в машинном словаре специального вектора соответствия признака записи и ее местоположения. Такой алгоритм можно получить на основе существующих способов интерполяции дискретного ряда значений. Однако следует отметить, что сам процесс вычисления адреса занимает относительно много времени, а для хранения коэффициентов интерполирующих полиномов требуется объем оперативной памяти, превышающий объем, обеспечивающий хранение кода признака.

Преобразование исходного признака в адрес можно выполнить, если рассматривать признаки и их адреса в виде полиномов [17]. В этом случае кодирование множества записей решается как задача его однозначного отображения на некоторое множество кодов, минимальное расстояние между которыми равно некоторой целочисленной величине d , т. е. кодовому расстоянию между двумя соседними признаками. При организации кодирования признаки представляются полиномом

$$N(x) = \sum_{i=1}^n a_i x^{i-1}, \quad (13)$$

где a_i — i -й признак;

x^{i-1} — коэффициент полинома.

Адрес хранения записи может быть представлен полиномом

$$R(x) = \sum_{i=1}^n b_i x^{i-1}, \quad (14)$$

где b_i — i -й адрес.

Значительно проще задача взаимно-однозначного соответствия для ограниченных условий решается в работе [22]. Задача формируется следующим образом: уста-

новить взаимно-однозначное соответствие между множеством всех сочетаний из N признаков по M и отрезком $\left[1, \left(\frac{N}{M}\right)\right]$ натурального ряда. Это соответствие задается таблицей, однако при значительном увеличении количества сочетаний такой способ реализации соответствия неприменим на практике из-за большого расхода оперативной памяти.

3.3.4. Редактирование информации

Редактирование информации, хранящейся в информационной библиотеке, составляет значительную часть обработки массивов в автоматизированных системах управления. Приемы редактирования показаны на рис. 3.30.

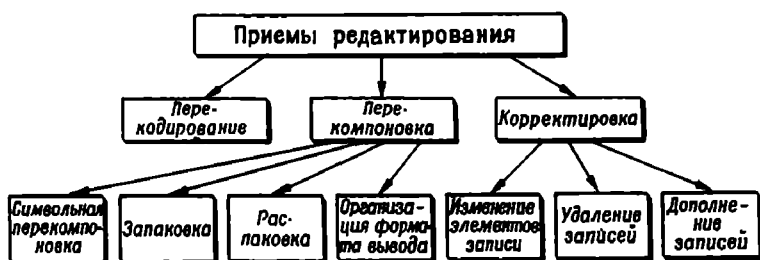


Рис. 3.30. Классификационная схема приемов редактирования

Наиболее распространенными приемами редактирования являются перекодирование, перекомпоновка и корректировка.

Перекодирование

Существующее многообразие процедур перекодирования обеспечивает преобразование кодов (символов, слов, записей или массивов), вызываемое различным представлением информации в устройствах ЭЦВМ.

Исходная и промежуточная информация военных задач может быть введена в память ЭЦВМ в восьмеричной или десятичной системах счисления (цифровая информация) и в текстовом виде (алфавитно-цифровая информация).

Процедуры перекодирования на ЭЦВМ „Минск-22М“

Вывод			Перфолента			Перфокарты	
			Цифровая информация		Алфавитно-цифровая информация	Цифровая информация	Алфавитно-цифровая информация
			2 с. с.	10 с. с.	2 МТК	Код карт (КК)	Код карт (КК)
Быстродействующий печатающий механизм	Код машины	8 с. с. 10 с. с.	— 2→10	10→2 —	2 МТК→2 2 МТК→10	КК→2* КК→10*	КК→2 КК→10
Перфоратор лент № 1	Код машины	2 с. с. 10 с. с. алфавитно-цифровой	— 2→10 2→2 МТК	10→2 — 10→2 МТК	2 МТК→2 2 МТК→10 —	КК→2* КК→10* КК→2 МТК	КК→2 КК→10 КК→2 МТК
Перфоратор лент № 2	2 МТК	алфавитно-цифровой	2→2 МТК	10→2 МТК	—	КК→2 МТК	КК→2 МТК
РТА-50	2 МТК	алфавитно-цифровой	2→2 МТК	10→2 МТК	—	КК→2 МТК	КК→2 МТК
УПЧ-22	Код УПЧ	алфавитно-цифровой	2→АЦПУ *	10→АЦПУ *	2 МТК→АЦПУ	—	—

Примечания: 1. Знаком * помечены перекодировки, реализация которых выполнена аппаратно и производится по командам, предусмотренным системой команд ЭЦВМ.

2. Цифры 2, 8 и 10 являются основаниями соответствующих систем счисления, а знаком → показано направление перекодирования.

3. 2 МТК — второй международный телеграфный код.

Вывод промежуточной и результатной информации из машины может быть выполнен в восьмеричной или десятичной системах счисления, во втором международном телеграфном коде (в любом другом фиксированном коде) или в коде алфавитно-цифровой печати (код УПЧ).

Внутренняя обработка информации выполняется в большинстве современных ЭЦВМ в двоичной системе счисления. Исключение составляет ряд отечественных ЭЦВМ, обладающих специальной аппаратурой десятичной арифметики.

В связи с различным представлением информации на вводе (выводе) и в устройствах машины, используемых в последующей обработке информации, возникает необходимость введения процедур перекодирования. Возможный набор и характер процедур перекодирования на ЭЦВМ «Минск-22М» представлен в табл. 6.

Дополнительно отметим, что при организации процедур перекодирования необходимо учитывать также и форму представления вводимой (выводимой) информации: фиксированную или плавающую запятую, целые числа.

Перекомпоновка

Перекомпоновка информации связана с преобразованием формы представления единиц информации в памяти машины для обеспечения удобств ввода, внутренней обработки и вывода. В рассматриваемом классе задач наиболее широко используются символьная перекомпоновка записей входных массивов, запаковка, распаковка и организация формата вывода.

Символьная перекомпоновка предусматривает последовательный просмотр списка символов исходного массива и компоновку составных единиц записей, структура которых задается системой идентификаторов начального описания.

Текстовая информация вводится в память машины символ за символом. По мере заполнения буферного регистра символами производится пересылка содержимого этого регистра в очередную ячейку памяти. Если при проектировании текстового ввода не предусматривались специальные усилия, то структура записи не будет со-

гласована со структурой (ячеечной) памяти. При этом возможен случай, когда в одной из ячеек памяти закончится очередной элемент записи и в ней же начнется со следующего символа другой элемент записи.

Перемещение и обработка информации в устройствах большинства ЭЦВМ производятся обычно в соответствии со структурой памяти. Поэтому перед обработкой введенного массива необходимо перераспределить его в памяти так, чтобы каждая запись начиналась с новой ячейки памяти. При данной перекомпоновке взаимное положение каждого символа относительно друг друга не изменяется, а производится лишь согласование структуры записи со структурой памяти, что обеспечивает существенное упрощение подготовки носителей информации к вводу в ЭЦВМ.

Выполним запись алгоритма символьной перекомпоновки в нотации алгоритмического языка АЛГЭК [14], допускающего описание и обработку текстовой информации. Исходный массив текста идентифицирован буквой A . Предполагается, что исходный текст может быть «вычищен» от символов, не содержащихся в словаре допустимых символов $СЛ$ при условии, когда логическая переменная L есть истина. Перекомпоновка выполняется в текстовый массив B . Вариант записи алгоритма имеет следующий вид:

```

начало целые  $M, L, N, D$ ; ввод ('канал 1',  $M, L, N, D$ );
    начало текстовый массив  $B$  [ $1 : M$ ] вид ' $C(L)'$ ';
текстовые  $A$  вид ' $C(N)'$ ,  $СЛ$  вид ' $C(D)'$ ,  $P$  вид ' $C(1)'$ ;
    целые  $i, j, k$ ; логическое  $L$ ;
    ввод ('канал 1',  $A, P, СЛ, L$ );  $M := k := 1$ ;
    для  $i := 1 : N$  цикл  $B[i] := ' \_ '$ ;
    для  $i := 1 : N$  цикл
        начало если  $A$  [элемент  $i$ ] =  $P$  то на  $M2$ ;
            если  $L =$  ложь то на  $M1$ ;
            для  $j := 1 : D$  цикл
                если  $A$  [элемент  $i$ ] =  $СЛ$  [элемент  $j$ ] то на  $M1$ ;
            на  $M3$ ;  $M1 : B[M$  элемент  $k] = A$  [элемент  $j$ ];  $k := k + 1$ ;
            если  $k \leq L$  то на  $M3$ ;
             $M2 : k := 1$ ;  $M := M + 1$ ;

```

М3: конец цикла просмотра символов исходного текста;
вывод ('канал 2', В, М — 1)

конец

конец;

Процедура запаковки информации обеспечивает компактную форму размещения элементов записи (или записей) для хранения и пересылок на основании известных (заданных) характеристик элементов или записей. Типичными примерами используемых методов запаковки являются следующие:

— включение двух и более элементов в одно или более машинных слов;

— запоминание только ненулевых элементов записи и приравнивание всех элементов, не подлежащих запоминанию, к нулю;

— замена последовательных нулей индикатором числа заменяемых нулей.

Процедура распаковки информации обеспечивает получение исходных данных из упакованных данных.

Процедура организации формата вывода обеспечивает управление работой машины для получения желаемого расположения элементов выводимых записей, подавления незначащих нулей с «подтяжкой» знака минус к первой значащей цифре выводимого числа и для деления выводимых массивов на листы стандартной длины при обязательном стандартном оформлении каждого листа (номер листа, шапка и соответствующие разделители листов).

Корректировка

Под корректировкой массивов понимается их текущее ведение, включающее изменения отдельных элементов записи, удаление и дополнение записей. От организации корректировки и алгоритмов ее реализации существенно зависит расход машинного времени на текущее ведение массивов, которое, по существу, обеспечивает подготовку этих массивов для решения конкретных задач автоматизированными системами управления. Учитывая большие объемы корректируемых массивов и степень их дублирования, можно утверждать, что сведение к минимуму времени подготовки массивов к работе является важной задачей.

Основной составной частью алгоритмов корректировки является поиск записей по заданному шифру или их совокупности. В связи с тем что поиск рассматривался ранее, в данном разделе обобщены возможные схемы организации корректировки информации (рис. 3.31).

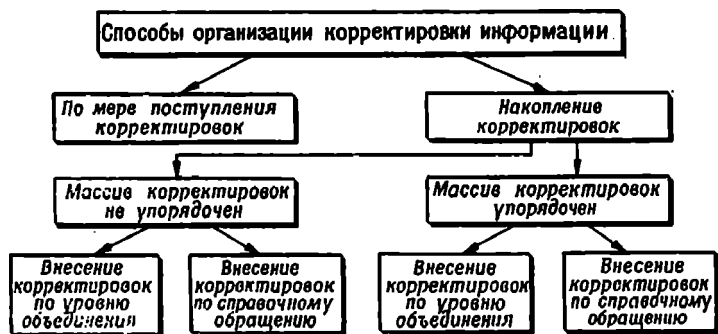


Рис. 3.31. Классификационная схема способов корректировки информации

Постановка задачи корректировки состоит в следующем. Пусть к некоторому моменту времени в рассматриваемый массив, представленный в информационной библиотеке n -дублями, были внесены все корректировки, имевшие место до этого момента времени. Такой массив назовем основным. Корректировки, поступающие после указанного момента времени, включаются в массив корректировок, объединяющий записи-заказы на корректировку.

Слияние основного массива и массива корректировок производится либо по мере поступления очередной записи-заказа, либо по уровню объединения или при наличии справочного обращения к основному массиву.

Внесение корректировок по мере поступления очередной записи-заказа характерно для массивов, используемых при решении задач сбора и оперативного анализа информации, например информации о средствах ядерного нападения противника. Действия по корректировке обычно выполняются в режиме прерывания других программ обработки данных за счет их более низкого приоритета по сравнению с приоритетом задачи корректировки.

Обычный режим работы по внесению корректировок предусматривает накопление записей-заказов. Как уже отмечалось, слияние основного массива с массивом корректировок может производиться также по уровню объединения, т. е. по достижении заданного количества записей-заказов в массиве корректировок или ограниченного отрезка времени до момента решения задач.

Если предполагается обращение за справкой к основному массиву, то вызывается к работе алгоритм корректировки основного массива при условии, что в массиве корректировок есть записи-заказы.

В процессе накопления массив корректировок может создаваться в упорядоченной последовательности относительно одного из дублей основного массива. Возможен также вариант накопления записей-заказов по мере поступления, т. е. когда массив корректировок не упорядочен относительно дублей основного массива. Заметим, что в большинстве случаев неупорядоченные массивы корректировок перед слиянием упорядочиваются в соответствии с функцией упорядоченности основного массива (очередного дубля основного массива).

Время упорядочения массива корректировок по признаку основного массива

$$t_y = 2t_{cp} \times M \left[\log_b \left[\frac{M}{S} \right] + 1 \right], \quad (15)$$

где t_{cp} — среднее время чтения (записи) одной записи с магнитной ленты с учетом разгона и перемотки ленты;

b — основание системы счисления, по которому производится внешняя сортировка информации;

M — количество записей массива корректировок;

S — количество записей в допустимой доле обмена информацией между оперативной и внешней памятью.

Время на слияние основного массива и массива корректировок

$$t_{cl} = 2t_{cp} (N + M), \quad (16)$$

где N — количество записей основного массива.

Отсюда машинное время, необходимое для слияния

упорядоченного основного массива из N записей и неупорядоченного массива корректировок из M записей,

$$t = 2t_{\text{ср}} \left(N + M \left[\log_b \left[\frac{M}{S} \right] + 2 \right] \right). \quad (17)$$

Задержка времени ответа в случае неупорядоченного массива корректировок по справочному обращению определяется зависимостью (15).

Заметим, что предварительное упорядочение массива корректировок при накоплении требует разработки специальных программ расстановки записей-заказов. Однако эти программы выполняют упорядочение только относительно одного из дублей основного массива, а упорядоченность относительно других дублей должна быть получена перед слиянием массивов. Усилия по упорядочению в процессе накопления записей (как временные, так и по разработке специальных программ) при большом числе дублей вряд ли могут быть оправданы.

Библиотечные программы редактирования ЭЦВМ «Минск-32»

Набор библиотечных программ редактирования показан на рис. 3.32.

Программа перекодирования алфавитно-цифровой информации из второго международного телеграфного кода в код ГОСТ 10859—64 обеспечивает перекодирование массива информации, размещенного в оперативной памяти по пять символов в ячейке, и заканчивается признаком конца информации. Для перекодирования используется таблица соответствия кодов. Так как в 2МТК используются три служебных регистра (русский, латинский и цифровой), то одной комбинации символа 2МТК соответствуют три комбинации символов кода ГОСТ 10859—64, одно из значений которых выбирается из таблицы в зависимости от служебного регистра, предшествующего перекодируемому символу. Результатный массив (по пять символов в ячейке) может быть получен как на месте исходного массива, так и в другом поле памяти.

Программа перекодирования числа, представленного в коде ГОСТ 10859—64, в десятичное целое число обеспечивает перекодирование текста по пять символов

в ячейке. Допустимыми символами исходного текста являются цифры, знаки плюс, минус и запятая. Если выделенный символ — цифра, то рабочая ячейка, в которой накапливается результат, сдвигается влево на четыре разряда, после чего выделенный символ приформировывается к младшим разрядам ячейки. Перекодирование завершается, если очередным символом является символ — признак конца текста.



Рис. 3.32. Структурная схема библиотечных программ редактирования ЭЦВМ «Минск-32»

Программа перекодирования десятичного целого числа в число, представленное в коде ГОСТ 10859—64, обеспечивает замену тетрад исходного числа символами данного кода. Ответ формируется в указанной последовательности ячеек по пять символов в ячейке. Знак плюс и незначащие нули числа опускаются. В конце последовательности записывается символ — признак конца числа.

Программа перекомпоновки обеспечивает перераспределение информации в оперативной памяти в целях изменения структуры записи. Исходным материалом для программы являются запись, расположенная в МОЗУ с заданного адреса, и таблица действий по перекомпоновке. Каждая ячейка таблицы характеризует положение

ние определенного элемента в исходной и в перекомпонованной записи. Программа выполняет функции распаковки или запаковки информации одной записи.

Программа вставки предназначена для вставки символа в число, каждая цифра которого представлена семиразрядным символом. Заданный при обращении символ вставляется в число, справа остается указанное количество символов. Остальные символы сдвигаются влево на символ.

Программа отбрасывания нулей предназначена для замены незначащих нулей в числе (знака в положительных числах) на заданный символ. Исходной информацией для программы являются число, расположенное в указанной ячейке памяти, и символ, на который необходимо заменить незначащие нули. После обработки число размещается в указанной ячейке памяти. Программа используется при формировании вывода алфавитно-цифровой информации на широкую печать.

3.4. СЕГМЕНТАЦИЯ БОЛЬШИХ МАССИВОВ, ХРАНЯЩИХСЯ ВО ВНЕШНЕЙ ПАМЯТИ

3.4.1. Общие сведения

В настоящее время подавляющее число задач, решаемых с помощью автоматизированных систем обработки данных, составляют задачи, имеющие следующие особенности:

- 1) блочный принцип организации программ обработки данных с последующим поэтапным делением блоков по информационному принципу;

- 2) зависимость хода дальнейшей обработки данных от значения некоторых параметров, получаемых в процессе счета;

- 3) информационную взаимосвязь по обрабатываемым массивам данных в большинстве задач АСУ;

- 4) большой объем исходных, промежуточных и выходных данных;

- 5) большое число массивов различных типов, принимающих участие в решении задач при наличии сложных схем взаимосвязей в соответствии с алгоритмом обработки данных;

- 6) размещение массивов обрабатываемых данных во

внешней памяти с учетом неоднократного просмотра в процессе решения задач;

7) необходимость решения задачи сегментации больших массивов данных при ограниченной оперативной памяти с учетом информационных взаимосвязей как в блоке, задаче, так и в АСУ в целом.

Любой этап (блок) обработки данных в этих задачах может быть описан с довольно высокой степенью адекватности в рамках следующей модели.

Имеется множество $N = \{1, 2, \dots, n\}$ массивов, подлежащих обработке в рамках конкретного этапа и располагающихся перед началом обработки во внешней памяти. Каждый из этих массивов характеризуется следующими параметрами: длиной M_i , выраженной в соответствующих единицах (словах, символах, байтах и т. д.), количеством записей d_i в массиве, длиной одной записи (фиксированной) m_i . При этом, естественно, предполагается, что $M_i = d_i m_i$. Входные массивы из множества N подвергаются просмотру и обработке в целях получения выходных массивов, обладающих аналогичными параметрами. При этом в процессе просмотра и обработки массивов производятся последовательное считывание входных массивов порциями в отведенные для них области ввода x_i оперативной памяти (сегменты) и последовательная запись порций выходных массивов из отведенных для них сегментов оперативной памяти во внешнюю память.

Общее время реализации некоторого этапа обработки данных на ЭЦВМ складывается из времени, затрачиваемого на выполнение следующих действий:

поиска местоположения массива (массивов) во внешней памяти;

последовательного считывания (записи) сегментов массивов в оперативную память или из нее;

собственно обработки сегментов данных, находящихся в оперативной памяти.

Время поиска местоположения массива во внешней памяти во многом определяется организацией хранения информации и может быть принято с точки зрения рассматриваемых вопросов за величину постоянную.

Время считывания (записи) порции массива зависит от объема области оперативной памяти, отведенной под размещение порции, и от первоначального распределе-

ния массивов во внешней памяти. Так, например, при размещении нескольких массивов на одной магнитной ленте возникает необходимость «бегать» по одной и той же ленте в поисках порции массивов. Однако при продуманном распределении массивов во внешней памяти таких случаев можно избежать или свести их к минимуму, так как количество массивов, обрабатываемых в пределах одного этапа, обычно меньше количества доступных блоков внешней памяти.

Рассмотрим такой выбор сегментов ввода (вывода), который позволил бы свести к минимуму общее время, затрачиваемое на последовательное считывание (запись) массивов. Естественно, что такой выбор не влияет на время собственно обработки, определяемое алгоритмом и средствами программного обеспечения, и собственно обработка не влияет на него. В общем виде задача сегментации массивов обрабатываемых данных в пределах этапа программы может быть сформулирована следующим образом.

Пусть имеется множество $N = \{1, 2, \dots, n\}$ массивов, размещенных во внешней памяти и характеризующихся длиной M_i , количеством записей d_i и длиной одной записи m_i . Требуется распределить свободную область Q оперативной памяти на порции ввода (вывода) S_i так, чтобы свести к минимуму общее число обращений к внешней памяти. Число обращений к внешней памяти при считывании (записи) i -го массива равно $\left\{ \frac{M_i}{S_i} \right\}$. Тогда задача сегментации может быть записана так:

$$\left. \begin{aligned} \sum_{i=1}^n \left\{ \frac{M_i}{S_i} \right\} &\rightarrow \min; \\ \sum_{i=1}^n S_i &\leq Q. \end{aligned} \right\} \quad (18)$$

Подобная постановка задачи не учитывает конструктивных особенностей внешней памяти существующих ЭВМ, которые порой существенно влияют на время и характер обращения к внешней памяти.

Так, например, для внешней памяти на МЛ ЭЦВМ «Минск-22» допускается прямая адресация каждого слова и возможен произвольный доступ к любому слову на

ленте. Подобная адресация позволяет записывать информацию на МЛ произвольными по длине сегментами, а затем читать ее сегменты, не совпадающие по длине с сегментами записи. Кроме того, наиболее перспективные внешние накопители типа магнитных дисков и магнитных барабанов большой емкости являются по сути дела также памятью с почти произвольным доступом.

Большинство современных ЭЦВМ («Днепр-21», АСВТ, РЯД, «Минск-32») имеет зонный принцип организации массивов на МЛ, позволяющий прочитать с магнитной ленты только записанную зону. Более того, нет возможности заменять выборочно отдельные зоны массива информации. Массив, полученный на МЛ, может многократно использоваться для чтения информации, а при необходимости выполнения корректировки массива он должен быть перезаписан на другую МЛ.

3.4.2. Сегментация для устройств со свободным доступом

Для решения выражения (18), сводимого к нахождению условного оптимума, может быть использован метод Лагранжа [18], в соответствии с которым запишем функцию $F(S_1, S_2, \dots, S_n, \lambda)$:

$$F = \sum_{i=1}^n \frac{M_i}{S_i} + \lambda \left(\sum_{i=1}^n S_i - Q \right). \quad (19)$$

Для нахождения оптимальных значений F необходимо решить систему $n+1$ уравнений с $n+1$ неизвестными:

$$\left. \begin{aligned} \frac{dF}{dS_i} &= 0 \quad (i = 1, 2, \dots, n); \\ \frac{dF}{d\lambda} &= 0. \end{aligned} \right\} \quad (20)$$

Для функции (19) система уравнений (20) принимает следующий вид:

$$\left. \begin{aligned} -\frac{M_i}{S_i^2} + \lambda &= 0 \quad (i = 1, 2, \dots, n); \\ \sum_{i=1}^n S_i - Q &= 0. \end{aligned} \right\} \quad (21)$$

Из первых n уравнений системы (21) следует, что

$$S_i = \sqrt{\frac{M_i}{\lambda}}.$$

Подставляя значения S_i в $n+1$ уравнение (21), получим

$$\frac{1}{\sqrt{\lambda}} \sum_{i=1}^n M_i - Q = 0$$

или

$$\sqrt{\lambda} = \frac{1}{Q} \sum_{i=1}^n \sqrt{M_i}.$$

Подставляя найденное значение $\sqrt{\lambda}$ в первые n уравнений системы (21), окончательно получим

$$S_i = \frac{\sqrt{M_i}}{\sum_{i=1}^n \sqrt{M_i}} Q. \quad (22)$$

Таким образом, для оптимального (т. е. суммарного количества обращений к внешней памяти) распределения свободной памяти под сегменты S_i ввода (вывода) необходимо брать пропорциональными квадратному корню из значений длин соответствующих массивов.

Описанный выше метод выбора сегмента предпочтительнее обычно используемого метода назначения сегментов, пропорциональных длинам массивов. Для случая, когда сегмент пропорционален длине массива, запишем:

$$S_i = \frac{M_i}{\sum_{i=1}^n M_i} Q. \quad (23)$$

При таком подходе общее количество обращений к внешнему накопителю можно определить из зависимости

$$K_1 = \sum_{i=1}^n \frac{M_i}{S_i} = \frac{\sum_{i=1}^n \sum_{i=1}^n M_i}{Q} = \frac{n}{Q} \sum_{i=1}^n M_i. \quad (24)$$

При назначении сегментов на основании зависимости (22) общее количество обращений составит

$$K_2 = \sum_{i=1}^n \frac{M_i}{S_i} = \frac{1}{Q} \left(\sum_{i=1}^n \sqrt{M_i} \right)^2. \quad (25)$$

Относительное уменьшение числа обращений к внешней памяти можно оценить по зависимости

$$K = \frac{K_2}{K_1} = \frac{\left(\sum_{i=1}^n \sqrt{M_i} \right)^2}{n \sum_{i=1}^n M_i}. \quad (26)$$

Анализ зависимости (26) при различных наборах n и M_i показывает, что в любом случае $K \leq 1$. При этом $K=1$ только тогда, когда $M_1=M_2=\dots=M_n$.

В результате анализа ряда задач [18] установлено, что в большинстве случаев количество массивов, обрабатываемых совместно, составляет $n=4$.

3.4.3. Сегментация для устройств с зонным доступом

В устройствах, в основе которых лежит зонный принцип организации внешней памяти, с учетом использования массивов на различных этапах решения многоэтапной задачи обработки данных, проблема выбора оптимальных сегментов оперативной памяти значительно усложняется. Усложняется эта проблема еще и тем, что при проектировании автоматизированных систем управления обычно предусматривается использование одних и тех же массивов в ряде задач обработки данных.

Для решения этой проблемы ставится следующая задача. Для некоторой подсистемы (системы) обработки информации программы, реализующие обработку данных, разбиваются на отдельные этапы. Этапы связаны между собой информационно и имеют некоторый приоритет выполнения, основанный на информационной преемственности. В каждом из выделенных этапов используются отдельные массивы, хранимые во внешней памяти с последовательным доступом. Эти массивы могут получаться в результате работы соответствующего этапа

(массив будет записываться во внешнюю память), а далее использоваться (будут считываться из внешней памяти) для получения новых массивов.

На каждом из этапов часть оперативной памяти занята программой данного этапа и резидентной частью операционной системы. Оставшаяся часть оперативной памяти может быть использована для сегментов, выделяемых массивам, участвующим в обработке на данном этапе.

При использовании в некоторых программах подсистемы массива данных, полученного в другой программе, свободную оперативную память этих этапов требуется уменьшить на величину сегмента (длину зоны) массива. Необходимо для всех массивов, используемых и получаемых в программах подсистемы, определить сегмент (длину зоны) при условии, что суммарное количество обращений к внешней памяти будет минимальным.

Формализованная постановка задачи состоит в следующем. Необходимо найти такие $S_1, S_2, \dots, S_n$, чтобы функционал

$$T(S_1, S_2, \dots, S_n) = \sum_{i=1}^n \alpha_i \frac{M_i}{S_i} \quad (27)$$

принимал минимальное значение при ограничениях:

$$\left. \begin{array}{l} \sum_{i \in P_1} S_i \leq Q_1; \\ \sum_{i \in P_2} S_i \leq Q_2; \\ \dots \dots \dots \\ \sum_{i \in P_k} S_i \leq Q_k; \end{array} \right\} \quad (28)$$

$$\left. \begin{array}{l} Q_j > 0, \quad j = 1, 2, \dots, k; \\ S_i \geq 1, \quad i = 1, 2, \dots, n; \\ S_i - \text{целые числа,} \end{array} \right\} \quad (29)$$

где α_i — количество использований (записей или чтений) i -го массива; очевидно, что $\alpha_i \geq 1$;

k — количество этапов в программе подсистемы.

Алгоритм решения сформулированной задачи сводится к выполнению действий в такой последовательности:

1. Выбирается начальное значение

$$S^0 = \{S_1^0, S_2^0, \dots, S_n^0\}.$$

В связи с тем что зависимость (27) является выпуклой вниз функцией и принимает максимальные значения на границе, можно выбрать в качестве начальных приближений $S_i^0 = h_i$, где h_i — длина записи i -го массива при $i = 1, 2, \dots, n$. Выбор записи в качестве минимальной единицы информации для распределения объясняется тем, что в обработке должна участвовать, как правило, вся запись.

2. Вычисляется значение функционала (27) в выбранной точке S^0 с получением

$$T_0 = f(S^0).$$

3. Из множества точек S_i ($i = 1, 2, \dots, n$) строится подмножество

$$N = \{S_j, f(S_j) < f(S^0)\},$$

удовлетворяющее заданным ограничениям.

4. Если подмножество N пусто, то точка S^0 является искомым решением.

5. Если подмножество N не пусто, то выбирается точка таким образом, чтобы

$$f(S_{i_0}) = \min_{j \in N} f(S_j).$$

6. Полученная точка S_{i_0} рассматривается далее в качестве исходной точки S^0 с повторением этапов 2—5-го данного алгоритма.

На основании ранее приведенных характеристик функционала (27) и ограничений (28) и (29) очевидно, что найденная после описанного направленного перебора точка будет искомым решением.

Необходимо признать, что путь последовательных приближений к искомому решению является сравнительно медленным и реализация этого алгоритма применительно к большому набору массивов данных АСУ немыслима без использования ЭЦВМ.

Глава 4

ОСНОВНЫЕ ПРИНЦИПЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМ ОБРАБОТКИ ВОЕННОЙ ИНФОРМАЦИИ НА ЭЦВМ

В предыдущей главе были рассмотрены основные способы и приемы обработки на ЭЦВМ больших массивов военной информации. Важно отметить, что этот рабочий арсенал инженера-программиста является обязательным для обработки информации в любой системе управления войсками, при любой степени автоматизации системы управления.

Анализ многочисленных задач обработки военной информации (ОТИ) показывает, что эти задачи сводятся к многократному выполнению немногих типовых процедур (см. гл. 3).

Для автоматизации процедур обработки больших массивов военной информации целесообразно разрабатывать комплексные проблемно-ориентированные системы, т. е. системы обработки военной информации на базе программного (математического) обеспечения ЭЦВМ и за счет его расширения.

Создание систем обработки обычно осложняется следующим:

- алгоритмизация и программирование выполняются специалистами с разным уровнем подготовки;
- создаваемые системы подвержены частым изменениям;
- эксплуатация подобных систем предполагает одновременное прохождение через систему потока взаимосвязанных задач, а это связано с изменением содержания и очередности выполнения работ;

— большинство систем обработки военной информации превращаются в «человеко-машинные» или в системы многовариантного моделирования реальных процессов военного управления;

— при проектировании систем трудно предвидеть обратное воздействие автоматизации на объем, состав и способы обработки информации. Поэтому необходимо предусматривать возможности роста системы и перехода на более новые и мощные ЭЦВМ.

Выделим следующие основные принципы создания систем обработки на ЭЦВМ информации:

1. Система должна иметь единые, централизованные массивы информации, состав которых должен описываться. Это дает свободный доступ к исходным данным и промежуточным результатам для всех задач и пользователей (абонентов) системы.

2. Система состоит из подсистем, количество которых должно соответствовать принятым этапам обработки военной информации в системе. Подсистемы собираются из автономных подпрограмм («модулей»), которые образуют иерархическую структуру.

3. При разработке подпрограмм необходимо применять принцип управляемости алгоритмов. Основные подпрограммы («модули») системы должны быть обобщенными и получать конкретное содержание по управляющей информации.

4. Система должна работать в режиме «человек — машина». Этот режим сводится к выполнению отдельных заданий пользователя (абонента). При этом доступ к системе должен быть на уровне «элементарных работ». Например, в заданиях на работу могут указываться массивы обрабатываемых данных и их основные параметры, применяемые массивы управляющей информации и т. п.

5. Система обработки должна быть трехуровневой системой, т. е. обеспечивать возможность работы с ней следующих специалистов:

практиков, решающих конкретные задачи обработки через задание и ввод в ЭЦВМ заявок;

программистов-проблемников, описывающих автоматизируемые задачи, т. е. готовящих управляющую информацию;

программистов-системников, расширяющих и развивающих программное обеспечение системы.

6. Система обработки должна быть открытой, т. е. допускать расширение и добавление «модулей» на новые работы.

7. Системы обработки проектируются обычно как проблемно-ориентированные системы на алгоритмических языках процедурного типа.

4.1. СОСТАВ И ХАРАКТЕРИСТИКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СОВРЕМЕННОЙ ЭЦВМ

Эффективная эксплуатация современных ЭЦВМ невозможна без оснащения их комплектом специальных программ, называемым программным обеспечением (ПО).

Системой программного обеспечения вычислительной машины (или комплекса машин) называется совокупность программ регулярного применения, описаний и инструкций по их применению, предназначенная для технической эксплуатации ЭЦВМ и ее использования.

Пользователь, ставящий задачи в крупном плане, в понятие программного обеспечения (ПО ЭЦВМ) включает все программы решения задач определенного класса и средства обеспечения процесса от постановки задач до их машинной реализации.

Те, кто разрабатывает и непосредственно применяет программное обеспечение, в это понятие включают все средства, предназначенные для обеспечения составления, отладки и выполнения программ на ЭЦВМ.

ПО ЭЦВМ разрабатывается в целях:

- упрощения общения оператора с машиной;
- сокращения сроков прохождения задач от их постановки до машинной реализации;
- повышения эффективности использования ЭЦВМ и многомашинных комплексов.

Исходя из задач и функций, выполняемых различными элементами ПО ЭЦВМ, можно выделить общее (внутреннее) и специальное (внешнее) программное обеспечение.

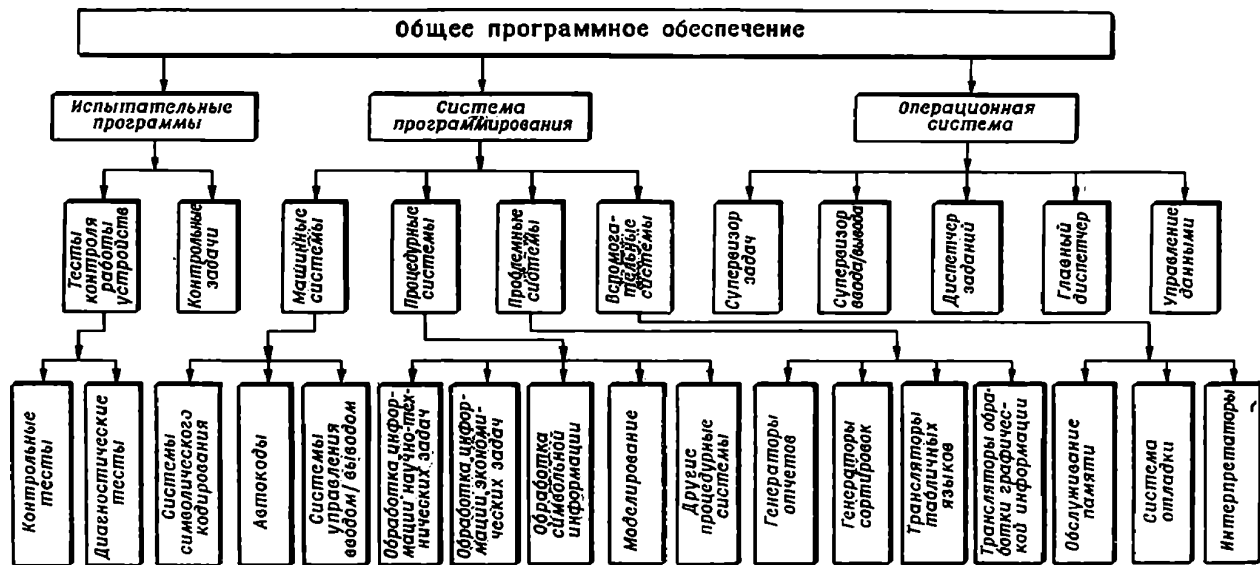


Рис. 4.1. Классификационная схема общего программного обеспечения ЭЦВМ

4.1.1. Общее программное обеспечение

Общее ПО позволяет использовать ЭЦВМ как некоторую универсальную систему обработки информации. Классификационная схема общего ПО ЭЦВМ показана на рис. 4.1.

По характеру использования общее ПО ЭЦВМ разделяется на три группы:

- испытательные программы для наладки, проверки и технической эксплуатации ЭЦВМ;
- система программирования и подготовки данных;
- операционная система.

Испытательные программы

В связи с широким распространением ЭЦВМ, естественно, встает вопрос об их эксплуатационной надежности. В частности, очень важными вопросами этой проблемы являются обнаружение факта появления отказа (с этим связан вопрос достоверности информации) и определение места неисправности для последующего восстановления работоспособности ЭЦВМ. Поэтому проектирование систем контроля стало одним из важных и перспективных направлений в общем комплексе мероприятий по повышению надежности функционирования ЭЦВМ и ряд «модулей» контроля включен в состав программного обеспечения.

Испытательные программы проверки работоспособности, наладки и технической эксплуатации ЭЦВМ принято разделять на контрольные задачи и тесты. Контрольные задачи предназначены для комплексной проверки работоспособности ЭЦВМ при приемно-сдаточных испытаниях, а также после проведения соответствующих профилактических работ (годовая и полугодовая профилактика). Тесты выполняются обычно по отдельным устройствам ЭЦВМ и предназначены для выявления их работоспособности. Тесты бывают контрольные, устанавливающие факт наличия неисправности, и диагностические, определяющие отказавший элемент схемы. Это позволяет выполнять проверку ЭЦВМ поблочно с помощью контрольных тестов, а диагностические тесты используются только при наличии неисправностей.

Система программирования

Система программирования и подготовки данных обеспечивает подготовку и отладку задач. Сущность системы программирования заключается в следующем. ЭЦВМ может воспринимать задачу только в том случае, когда последняя записана в виде детального перечня команд на машинном языке, значительно отличающемся от обычного математического языка. Система программирования должна освободить программиста от необходимости изложения задачи на машинном языке путем введения специального языка более высокого уровня и более близкого к естественному языку человека. Таким образом, элементарная система программирования должна включать по меньшей мере два компонента:

- специальный язык (входной язык системы программирования) описания задач;
- программу, обеспечивающую перевод задач со специального языка на машинный.

Система программирования позволяет снизить трудоемкость той части разработки программ для ЭЦВМ, которая связана непосредственно с написанием программы. Это объясняется следующим:

- входной язык системы программирования ближе к естественному языку человека, что избавляет программиста от необходимости изучения формализованного машинного языка;

● изобразительные средства входного языка позволяют осуществлять более компактную запись алгоритмов обработки информации, чем это можно сделать на конкретном машинном языке, т. е. уменьшается объем ручной работы программиста;

- уменьшается количество технических ошибок программирования в силу того, что входной язык позволяет делать укрупненную запись вычислительных схем на основе принятых соглашений в этом языке;

● сокращается время на отладку программ, так как большинство существующих систем программирования располагает совершенным аппаратом поиска как логических, так и синтаксических ошибок в программах, записанных на входном языке;

- ряд систем программирования вообще исключает какую-либо работу по изложению алгоритма обработки,

кроме указаний на характер обрабатываемой информации и ее структуру; например, для генератора отчетов необходимо указать только форму входной и выходной информации, а программа обработки информации генерируется системой программирования.

В соответствии со структурой, уровнем формализации входного языка и целевым назначением систем программирования принята следующая их классификация [42]: машинно-ориентированные, процедурно-ориентированные, проблемно-ориентированные, вспомогательные системы.

Машинные системы программирования

Машинные системы программирования имеют входной язык, связанный с определенной машиной или семейством машин, и не могут быть приспособлены к иному виду машин.

Наиболее типичными представителями такой системы программирования являются системы символического кодирования, автокоды (макрокоды), системы управления вводом — выводом.

Системы символического кодирования (ССК) являются одними из первых систем, созданных для автоматизации программирования с использованием входных языков уровня «один к одному». Этот уровень предполагает, что из одной символической инструкции строится одна машинная команда или константа. Смысл применения данной системы состоит в использовании символической нотации вместо машинной, в применении автоматического распределения памяти и присвоении действительных адресов.

Системы символического кодирования являются базой для более совершенных систем автоматизации программирования по следующим причинам:

- входной язык высшего уровня транслируется в символический, а машинная программа получается затем в результате работы транслятора ССК;

- если ЭЦВМ имеет особенности устройства, которые не могут быть отражены в процедурной системе, то целесообразно перейти от процедурного языка к символическому, чтобы эффективнее использовать эти особенности машины.

Кроме этого, ССК имеет и самостоятельное значение как система программирования малых ЭЦВМ, для которых нельзя создать транслятор с языков высшего уровня в силу ограниченных возможностей машин.

Система символического кодирования включает следующие составные части:

- входной язык символического кодирования для записи программ и их изменений в удобной и наглядной форме;
- программу-транслятор, обеспечивающую контроль и перевод исходной программы в машинную программу;
- библиотеку стандартных подпрограмм на машинном языке;
- обслуживающие программы системы;
- описания языка и инструкции по использованию системы символического кодирования и порядку проведения работ по программированию, перфорации, трансляции и т. д.

Являясь машинно-ориентированным, язык символического кодирования требует от программиста знания основных приемов непосредственного программирования и позволяет ему в полной мере проявить искусство для написания эффективных программ.

Дальнейшим развитием автоматизации программирования на уровне машинно-ориентированных систем является использование автокодов. Входным языком автокодов является язык уровня «один к нескольким», т. е. из одной автокодовой инструкции строится несколько машинных команд. Основу автокодового языка составляют макрокоманды, обозначающие функцию или процедуру одной записью.

Макрокоманды переводятся в машинные команды одним из двух машинных путей: подстановкой или генерированием.

В подстановочной системе каждой макрокоманде соответствуют специальные библиотечные модули на машинном языке, реализующие действия, предусматриваемые макрокомандой. Для настройки модулей на конкретную работу в макрокоманде должны быть указаны параметры очередного обращения, по которым транслятор и выполняет настройку.

Генерирующая система содержит специальные программы, анализирующие макрокоманды. При анализе

выясняется, какую функцию требует исполнить программист, и формируются команды, реализующие эти функции.

Подстановочные и генерирующие системы содержат транслятор с символического языка. Поэтому программист при составлении программ может одновременно употреблять как символические команды, так и макрокоманды.

По сравнению с ССК автокоды имеют следующие преимущества:

- улучшенную внутреннюю организацию в целях ускорения трансляции;
- наличие разнообразных макрокоманд вывода, обеспечивающих вывод на большинство внешних устройств в форме, пригодной для последующего ввода;
- меньшие трудовые затраты на составление программ за счет применения макрокоманд;
- применение более совершенных методов обнаружения ошибок в исходных программах на этапе трансляции;
- сокращение времени прохождения задач от их постановки до машинной реализации.

Макроязык обычно конструируется по образцу и подобию системы команд конкретной ЭЦВМ и поэтому лишен универсальности, свойственной языкам высшего уровня. Именно в силу этого обстоятельства, а также из-за приспособленности к логической структуре определенной ЭЦВМ возможно построение более простых и компактных трансляторов с макроязыков автокодов. Кроме того, программы, составленные с помощью автокодов, по своему качеству в большей степени приближаются к программам, составляемым вручную программистом средней квалификации, отличаясь от последних не более чем на 10—15% по объему и времени решения задач.

Одной из специфичных систем программирования, ориентированных на конкретные ЭЦВМ, является система ввода — вывода (СУВВ). Совершенствование устройств ввода — вывода ЭЦВМ ведется по пути повышения скорости передачи информации, возможности совмещенного (параллельного) выполнения операций чтения, записи и вычислений, прерывания с учетом приоритетов устройств и разработкой специальных устройств

автоматического обнаружения ошибок. В связи с этим постоянно усложняется программирование работы устройств ввода — вывода.

В целях более полного использования возможностей ввода — вывода и упрощения программирования этих операций ЭЦВМ сопровождаются системами управления вводом — выводом. Основные функции СУВВ:

- одновременное управление всеми устройствами и составными элементами системы ввода — вывода;

- сбор данных об объектах, подлежащих вводу — выводу, для оптимизации работы ЭЦВМ;

- разработка структуры записей для хранения информации;

- создание и текущая проверка стандартных меток таких внешних носителей информации, как магнитные ленты, барабаны и диски;

- предварительная настройка элементов системы ввода — вывода на заданную работу в целях сокращения времени обмена и повышения эффективности использования ЭЦВМ;

- управление вводом — выводом с учетом приоритета объектов управления (массивов данных);

- выявление и исправление ошибок, появляющихся при выполнении ввода — вывода;

- прерывание работы рабочих программ в контрольных точках с запоминанием информации, хранящейся в оперативной памяти, при перерывах в работе;

- возобновление вычислительного процесса с контрольной точки прерывания.

Выполнение СУВВ перечисленных выше функций позволяет освободить программиста от трудоемкой работы по программированию деталей ввода — вывода и дать ему возможность больше думать не о форме, а о логике решаемой задачи. Вместо детальной программы ввода — вывода пишутся лишь команды обращения к СУВВ, которая обеспечивает реализацию требуемого объема работ.

Объем и логическая сложность СУВВ зависят от возможностей ЭЦВМ и особенностей обслуживаемых задач. Одним из важнейших требований, предъявляемых к конструированию СУВВ, является обеспечение модульного принципа ее строения. Это означает, что отдельные блоки системы обладают известной автономностью и про-

граммист может пользоваться произвольным набором блоков, необходимых для его программы. СУВВ используется как самостоятельно, так и в составе операционной системы.

Процедурные системы программирования

В отличие от машинно-ориентированных систем программирования процедурные системы используют в качестве входного языка для описания алгоритмов задач различные алгоритмические языки, не зависящие от конкретных ЭЦВМ. Эти языки, относящиеся к уровню «один к нескольким», выгодно отличаются от автокодов тем, что полностью освобождают программиста от записи программ, похожих на машинные.

Основными компонентами процедурных систем программирования являются входной алгоритмический язык и компилятор с него на уровень ССК или на уровень непосредственно машинных команд. В соответствии с целевым назначением алгоритмических языков можно рассматривать системы программирования, ориентированные на описание и обработку информации научно-технических и инженерных задач (АЛГОЛ, ФОРТРАН, и другие), экономических задач (КОБОЛ, АЛГЭК, АЛГЭМ и другие), символической информации (ЛИСП, ЭПСИЛОН и другие), моделирования (СИМУЛА) и другие системы, базирующиеся на конкретных алгоритмических языках.

Применение процедурных систем программирования позволяет в значительной мере сократить время прохождения задач от момента постановки до машинной реализации, обеспечивая повышение производительности труда программистов. Однако время счета по программам, полученным в результате трансляции, в определенной мере зависит от качества компилятора и даже в лучших разработках значительно превышает время счета по программам, полученным с использованием ССК или автокодов.

В нашей стране наиболее распространенными процедурными системами являются системы, использующие алгоритмические языки АЛГОЛ, КОБОЛ, ФОРТРАН и АЛГЭК.

Проблемные системы программирования

Проблемные системы программирования исключают какую-либо работу программиста по разработке алгоритмов решаемых задач. Эти системы ориентированы на узкий класс однотипных задач. Так, например, одна из первых проблемных систем программирования — генератор отчетов — была предназначена для подготовки программ, составляющих отчет по совокупности входных документов. В задачу программиста при работе с генератором отчетов входит задание информации о структуре и местоположении входного массива информации, формате и структуре требуемого отчета. При этом имеется в виду, что алгоритм получения отчета не описывается.

Существуют две разновидности генераторов отчетов. Первая разновидность предусматривает наличие инвариантных программ, настраивающихся по информации о входных массивах и отчете. После выполнения настройки такой генератор обеспечивает ввод исходного массива и изготовление отчета.

Вторая разновидность на базе информации о входных массивах и отчете генерирует рабочую программу, выполнение которой обеспечивает получение отчета заданного вида.

Для той и другой разновидности генераторов отчетов требуется задание следующих групп информации:

- описание ввода, включающее описание типов входных документов и признака, по которому различается тип документа;
- описание данных, включающее описание элементов информации, составляющих исходную запись, их выделение для реализации операции, а также описание процессов получения сумм элементов исходных записей;
- описание вычислений, включающее описание вычислительных схем для получения показателей отчета на основе исходных данных;
- описание вывода, включающее определение формата отчета с учетом заголовка, шапок, нумерации страниц и других особенностей оформления выходного документа.

Естественно, что в подобных системах программиро-

вания вопросы отладки программ практически исключены, что в значительной мере сокращает время от постановки задачи до ее машинной реализации. Однако необходимым элементом генераторов и других проблемных систем программирования является контроль за соблюдением условий задания исходной информации.

Генераторы сортировок обеспечивают выбор оптимальных алгоритмов упорядочения информации в соответствии с заданным законом на основании параметров сортируемых массивов (количество сортируемых записей, длина записи, длина признака сортировки).

В структурном плане генератор сортировок включает управляющую программу и набор модулей, реализующих различные алгоритмы сортировки информации. Управляющая программа по заданным параметрам сортируемого массива обеспечивает подключение одного из модулей, а после завершения упорядочения обеспечивает вывод полученного массива на заданный носитель информации (магнитные ленты, перфокарты и т. д.).

Одним из элементов проблемных систем программирования являются системы, базирующиеся на табличных языках или таблицах принятия решений (ТПР).

Идея, положенная в основу такой системы, состоит в том, что процесс обработки информации и сама обрабатываемая информация описываются с помощью ТПР. Соответствующим образом закодированные ТПР вводятся в память машины, транслируются в рабочую программу с ее последующим решением.

В пользу табличных языков свидетельствуют те обстоятельства, что ТПР независимы от машины, исключают словесное описание задачи, составление блок-схемы и программы. Кроме того, ТПР являются удобным средством общения различных специалистов, работающих над общей проблемой.

Одним из направлений в дальнейшем расширении использования ТПР является создание простых и гибких препроцессоров с табличного языка на один из процедурных языков программирования, которые и обеспечивают получение рабочих программ.

Специфика табличных языков, ограничивающих круг решаемых задач, сдерживает их широкое распространение.

В комплект ЭЦВМ могут быть включены специаль-

ные устройства, обеспечивающие ввод графической информации. Особенности применения таких устройств, а также специфика работы с графической информацией требуют включения в состав программного обеспечения специальных проблемных систем, содержащих языки описания и трансляторы построения рабочих программ для работы с графической информацией.

Вспомогательные системы программирования

Три перечисленные выше системы программирования имеют в своей основе язык и используются для составления программ. Вспомогательные системы включают набор заранее подготовленных программ, выполняющих ряд вспомогательных функций при обработке данных. Такие компоненты вспомогательных систем программирования, как обслуживание памяти, системы отладки, моделирующие программы, используются в совокупности с различными компонентами машинных, процедурных и проблемных систем программирования для обслуживания работы ЭЦВМ.

Обслуживающая программа может называться системой программирования при соблюдении условия универсальности, т. е. способности настраиваться на обработку различных входных массивов информации. В работе любой обслуживающей программы можно выделить два этапа. На первом (начальном) этапе осуществляются прием управляющей информации и автоматическая настройка на обработку по заданным параметрам входных массивов. На втором этапе реализуется выполнение заданной программы.

Обслуживание памяти

Среди программ, обслуживающих память, можно выделить программы обслуживания магнитных лент, включающие следующий набор модулей:

- очистка и разметка лент;
- присвоение идентификаторов и начальных блоков лентам;
- перепись с магнитной ленты на перфокарты;
- перепись с перфокарт на магнитные ленты;

- документирование содержимого магнитных лент на печать;

- контроль за сохранностью информации на магнитных лентах.

Для программ, обслуживающих дисковую память, к перечисленному выше набору модулей следует добавить модули переписи с дисков на магнитные ленты и с магнитных лент на диски.

Программы обслуживания оперативной памяти обеспечивают разнообразные операции над содержимым памяти: загрузку рабочих программ в память и настройку их к работе в определенных условиях, вывод программ из памяти, очистку памяти.

Система отладки

Программы этой группы обеспечивают проверку и выявление ошибок в рабочих программах. Существенной особенностью средств отладки является то, что их использование не требует никаких знаний о рабочей программе, получаемой с помощью транслятора, или об инвариантной программе, получаемой автоматической настройкой по исходной информации.

Средства отладки разделяются на две категории: одни могут быть использованы в процессе выполнения рабочих программ, а другие в тех случаях, когда в программе в процессе выполнения произошла ошибка.

Средства отладки, используемые в процессе выполнения программы, обеспечивают вывод на печать промежуточной информации, документируют ход выполнения программы и выдают в явном виде результаты решения задачи.

Как бы щедро ни включались в программу описанные выше средства отладки, очень часто ошибки могут произойти в тех частях программы, которые либо не были обеспечены контрольной печатью, либо содержали недостаточное количество контрольных данных. Средствами обнаружения ошибок являются программы полной или частичной прокрутки рабочих программ, блокировки выполнения программ с выдачей на печать содержимого отдельных массивов промежуточной информации и разгрузки памяти при выходе на контрольные точки.

Для современных систем ПО ЭЦВМ характерно использование автоматических отладочных систем. Такие системы управляют работой основной программы и ряда обслуживающих программ, необходимых при отладке. Автоматические отладочные программы позволяют вести отладку без участия программиста.

Подобные системы (операторной отладки) значительно повышают эффективность отладки рабочих программ.

Интерпретаторы

Общий процесс интерпретации заключается в чередовании программирования и счета отдельных частей задачи на ЭЦВМ. Режим интерпретации предполагает наличие исходной программы, написанной на специальном языке, и набора модулей, осуществляющих трансляцию отдельных конструкций языка и их выполнение. Таким образом, интерпретаторы читают и выполняют операторы языка.

Известны следующие случаи использования интерпретаторов:

- моделирование реального вычислителя, т. е. моделирование на реальной машине операций существующей или разрабатываемой ЭЦВМ;

- симуляция гипотетической машины в целях изучения приемов ее использования, например симуляция обработки списков на «несимвольной» машине;

- интерпретация языков моделирования, предназначенных для описания параллельных процессов, на последовательных машинах;

- интерпретация последовательности управляющих инструкций в целях управления групповыми процессами обработки, процессами реального времени и разделением времени;

- интерпретирующие трансляторы, обеспечивающие выполнение программ на исходных языках непосредственно через интерпретатор.

Интерпретаторами часто пользуются, когда проектируется новая машина и для нее создается новая система программирования.

Практика проектирования интерпретаторов показывает, что их трудно писать, отлаживать и модифицировать. Кроме того, интерпретаторы обладают малой ско-

ростью работы, требуют постоянного присутствия в поле оперативной памяти вместе с интерпретируемой программой и сравнительно велики по объему.

Операционная система

Операционная система (ОС) представляет собой совокупность приемов и процедур для работы на машине; обеспечивает управление оборудованием ЭЦВМ, счет и прохождение задач, обмен с внешней средой и архивом, а также связь с оператором.

Учитывая сложную структуру ОС, многогранность решаемых задач и отсутствие систематизированных описаний, рассмотрим подробнее эти вопросы в специально выделенном разделе (4.2).

4.1.2. Специальное программное обеспечение

Специальное программное обеспечение состоит из комплектов (пакетов) программ, добавляемых к общему ПО ЭЦВМ. Эти программы включаются в операционную систему и обеспечивают функционирование ЭЦВМ (комплекса машин) как некоторой специализированной системы обработки информации. Отличительными особенностями компонентов специального программного обеспечения является ориентация на сравнительно узкий круг решаемых задач и большое их разнообразие.

В соответствии с назначением пакетов программ принята их классификация, показанная на рис. 4.2.

Пакет состоит из комплекса программ и сопроводительной документации. Системная организация пакета зависит от выбранной структуры пакета, которая может быть простой или сложной.

Простая структура представляет собой организацию пакета по типу библиотечных программ. Такой пакет состоит из набора программных модулей, выполняющих конкретную работу, и обслуживающих программ, осуществляющих редактирование и текущее ведение модулей.

Структура сложного пакета программ определяется совокупностью используемых программных средств внутреннего управления и обслуживания и включает:

- управляющую программу пакета;

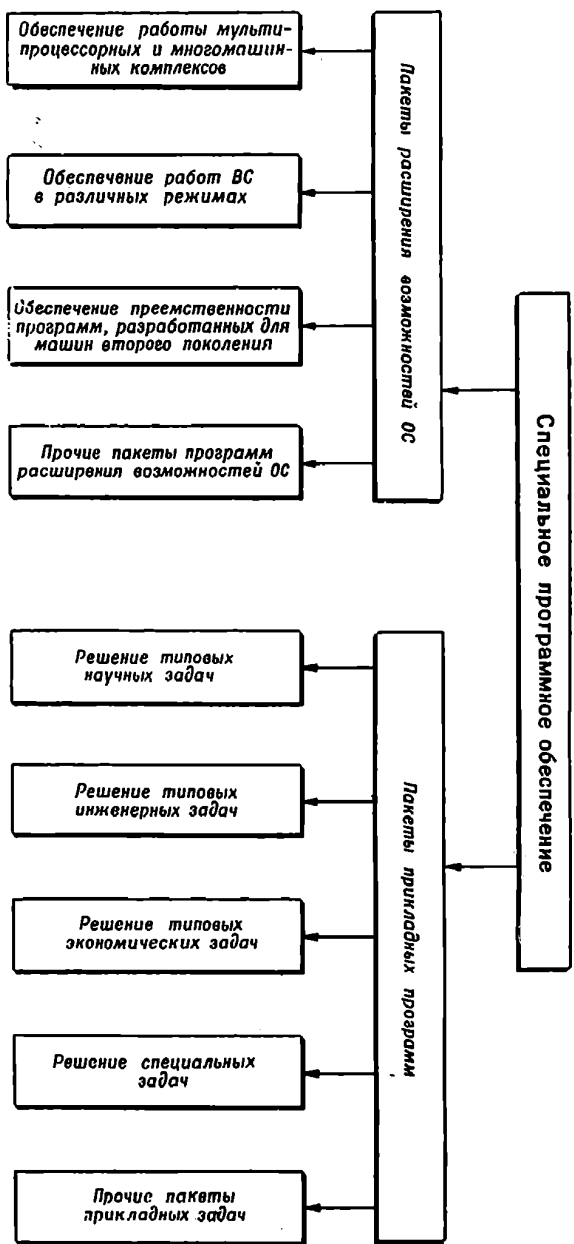


Рис. 4.2. Классификационная схема специального программного обеспечения ЭЦВМ

- процессор (транслятор) с входного языка;
- набор программных модулей, составляющих тело пакета;
- набор обслуживающих программ.

Управляющая программа обеспечивает управление общим ходом работы программных модулей пакета. Она работает под непосредственным управлением операционной системы ЭЦВМ (комплекса машин).

Процессор обеспечивает обработку программ действий, записанных на входном языке пакета.

Тело пакета состоит из набора программных модулей, предназначенных для решения определенного круга задач. Каждый программный модуль представляет собой программу (процедуру), реализующую алгоритм вычислительного или логического характера.

Набор обслуживающих программ включает ряд программ, выполняющих вспомогательные функции (средства редактирования и текущего ведения программ пакета, отладки и диагностики ошибок) и обеспечивающих определенные удобства пользователю при работе с пакетом.

Пакет, ориентированный на решение определенного класса задач, может допускать выделение разных уровней системной организации для обеспечения удобств использования при возможности применения на различных совместимых моделях ЭЦВМ.

При разработке пакетов программ могут быть выделены следующие уровни языков:

- языки написания программ пакета, являющиеся языками операционной системы, с которой должен работать пакет;
- язык, управляющий заданиями для ввода программ пользователя и их инициирования;
- входной язык пакета, предназначенный для задания пользователем параметров и управляющей информации в целях настройки программ пакета на решение конкретной задачи.

Документация по пакету программ должна содержать описания и руководства, необходимые для эксплуатации, редактирования и расширения возможностей пакета программ.

4.2. ОПЕРАЦИОННЫЕ СИСТЕМЫ

При эксплуатации вычислительных систем возникает проблема эффективного использования имеющегося оборудования. Понятие эффективности не определено и зависит от целей, средством достижения которых является вычислительная система (ВС).

Среди показателей, определяющих эффективность ВС в конкретных ситуациях, можно прежде всего отметить степень загрузки оборудования ВС. При прочих равных условиях чем выше загрузка оборудования, тем эффективнее организована работа ВС. Возможность быстрого решения той или иной задачи на данной ВС является также показателем эффективности. Одной из самых серьезных трудностей, связанных с решением задач обработки информации, является трудность подготовки задания и программ. Поэтому важным показателем организации работ на ВС является совокупность услуг, предоставляемых пользователю для упрощения и облегчения постановки задач и составления программ.

Следует отметить, что указанные выше показатели в определенном смысле находятся в противоречии друг с другом. Сохранение возможности быстрого решения отдельной задачи несовместимо с требованием максимальной загрузки всего оборудования ВС, так как трудно представить такую задачу, которая использовала бы все оборудование на протяжении всего времени обработки информации. Следовательно, для более полного использования оборудования желательно параллельное решение нескольких задач, при котором каждой из задач выделяется часть времени процессора. В то же время увеличение числа и объема услуг, оказываемых пользователю, приводит к снижению скорости обслуживания и к росту затрат вычислительных ресурсов на цели обслуживания.

Самой важной чертой системы обработки информации на ВС третьего поколения, существенно определяющей ее эффективность, является наличие операционной системы, представляющей собой организованную совокупность приемов и процедур для работы на машине.

Операционные системы обязаны своим возникновением, во-первых, развитию вычислительных средств и методов их использования и, во-вторых, усложнению и

увеличению состава программного обеспечения. Основной целью создания таких систем является приспособляемость к условиям различных применений и способов работы. Среди других целей, важность которых нельзя отрицать, следует выделить увеличение производительности ВС, повышение реакции системы, адаптируемость к изменяющимся ресурсам, возможность расширения системы и повышение производительности обслуживающего персонала.

Увеличение производительности современных ВС [44] обеспечивается тремя способами:

при управлении потоком работ, запущенных ранее для исполнения, ОС помогает оператору в выполнении ряда подготовительных действий;

система позволяет задачам, относящимся к различным заданиям, параллельно использовать ресурсы ВС в мультипрограммном режиме, что исключает необоснованные простои;

система располагает разнообразным набором программных языков и средств отладки.

Повышение реакции системы предполагает уменьшение времени ответа с момента поступления запроса до завершения действий, инициируемых этим запросом, применительно к различным режимам функционирования ОС.

Под адаптируемостью понимается приспособляемость к заданной конфигурации ВС и возможностям ОС, т. е. ОС должна быть рассчитана на многообразие применений и конфигураций оборудования.

С точки зрения возможности расширения ОС должна быть «открытой» для расширений и изменений, предполагающих добавление новых и исключение некоторых старых компонентов, что обеспечивает увеличение области применения вплоть до ввода новых режимов функционирования. Это свойство ОС предполагает использовать ее с новой аппаратурой, для новых областей применения и с новыми программами по мере того, как они появляются.

Повышение производительности труда обслуживающего персонала (программистов) может достигаться путем включения в систему относительно большого количества языков программирования и компактных языков управления заданиями, которые предназначены для оформления представляемых на машину работ.

4.2.1. Общие принципы построения ОС

Операционная система предъявляет единые требования ко всему программному обеспечению, разрабатываемому в рамках данной системы. Эти требования определяются основными функциями ОС:

- организацией связи и взаимодействия различных компонентов операционной системы;

- созданием, изменением и хранением информации пользователей системы;

- реализацией выбранного способа использования оборудования ВС и организацией потока задач.

Основные усилия при разработке ОС направляются на повышение пропускной способности системы обработки информации за счет следующих мероприятий:

- создание в ВС гибкой организации прохождения потока задач;

- простое и стандартное оформление заданий для ВС;

- обеспечение равномерной загрузки оборудования и других ресурсов системы;

- организация хранения в системе больших массивов информации стандартными способами при наличии сравнительно большего числа форм доступа к этой информации.

В зависимости от уровня развития ОС можно выделить несколько различных вариантов организации потока задач, преследующих основную цель — обеспечение эффективной загрузки оборудования [44]:

- пакетная обработка;

- пакетная обработка с мультипрограммированием;

- разделение времени.

Пакетная обработка предусматривает такую организацию прохождения задач, при которой на входе системы имеется пакет задач, последовательно пропускаемый через машину. Существенной чертой пакетной обработки является то, что каждая задача имеет в своем распоряжении почти все ресурсы системы (за исключением памяти, занятой резидентной частью).

Организацию прохождения потока задач берет на себя в основном оператор. Роль ОС (точнее, ее элементов) сводится к контролю задания пользователя, пуска задачи, выдачи необходимой информации о ходе решения

и снятии задачи с машины с последующим пуском очередной задачи пакета.

Допускается пакетная обработка с совмещением на вводе и выводе. Реализация функции ввода — вывода в этом случае передается ОС, которая оформляет их в качестве своих задач. Вывод информации при отсутствии свободных внешних устройств может производиться под управлением ОС на специальную ленту разгрузки. При освобождении устройств вывода информация с ленты разгрузки выводится на эти устройства под управлением ОС.

Ни одна задача пользователей при пакетной обработке не имеет преимуществ. Предполагаются различия только между задачами ОС и задачей пользователя. В этой связи недостатком режима пакетной обработки является относительно большое время реакции на запрос пользователя с момента передачи задачи в обработку до получения результатов решения.

Дальнейшим развитием пакетной обработки является введение режима мультипрограммирования с фиксированным числом задач, предполагающего совмещение нескольких независимых программ пользователей в единой ВС. Число одновременно решаемых задач пользователей в подобных системах определяется исходя из возможностей ВС, так, например, для ОС ЭЦВМ «Минск-32» и ИБМ/360 допускается совместное решение до четырех задач пользователей.

Выполнение задач пакета в режиме мультипрограммирования задается оператором в момент загрузки путем установления приоритета, определяющего порядок назначения ресурсов и выполнения задач. В системе создается входная очередь работ, из которых очередная программа задачи, получив управление, выполняется до прерывания, возникающего как из выполняемой программы, так и вне ее. После оценки ситуации управляющая программа решает вопрос выбора очередной запускаемой программы. Обычно управление получает программа, относящаяся к активной задаче с наибольшим приоритетом.

Наличие в оперативной памяти нескольких задач позволяет в более широком диапазоне решать проблему совмещения работы центрального процессора с операциями ввода — вывода. В этом случае задачи с низким

приоритетом образуют фон для задач с высоким приоритетом и могут получить необходимые массивы ввода — вывода только через системные программы. Задачи с большим приоритетом могут использовать для операций ввода — вывода как системные программы, так и непосредственный доступ, инициируемый в самой задаче.

Время реакции системы на запрос пользователя в значительной мере зависит от приоритета конкретной задачи, а также от приоритетов других задач пакета. Пропускная способность системы зависит от характеристик задач, выполняемых на каждом приоритетном месте.

Наиболее развитым режимом пакетной обработки является режим мультипрограммирования с переменным числом задач. Число одновременно присутствующих в оперативной памяти задач определяется состоянием динамически распределяемой части памяти и наличием других распределяемых ресурсов.

В системе организуются входная и выходная очереди работ, в которых задачи выстроены в соответствии с приоритетами. Приоритет задачи может назначаться оператором и уточняться ОС с точки зрения конкурентоспособности задачи на распределяемые ресурсы. Задача может создавать подзадачи, которым выделяются либо наличные ресурсы данной задачи, либо подзадачи требуют ресурсы у ОС.

В связи с тем что число задач переменное, борьба за ресурсы становится особенно жестокой, т. е. задачи с большим приоритетом могут отобрать ресурсы у задач с более низким приоритетом. При отсутствии задач с низким приоритетом ожидается освобождение ресурсов от своих подзадач или других задач. Оператор может вмешаться в порядок выполнения задач путем изменения приоритета.

Разделение времени является специфическим случаем мультипрограммирования, при котором некоторое число пользователей имеют одновременный доступ к общей системе (системе коллективного пользования). В системах разделения времени (СРВ) основным критерием оптимального планирования прохождения потока задач является уже не эффективное использование оборудования ВС, а обеспечение наименьшего времени обслуживания при ограниченном времени ожидания.

Механизм разделенного времени обеспечивает одновременное выполнение нескольких задач путем выделения каждой из них определенного кванта времени центрального процессора. Обслуживание задач производится из очереди, организованной в порядке поступления задач. Если очередной задаче недостаточен выделенный квант времени, то по его истечении эта задача становится в конец очереди.

Для повышения коэффициента загрузки оборудования в режиме СРВ в качестве фона может быть использован пакет несрочных задач.

Любая работа для выполнения на ВС должна быть оформлена в соответствии со стандартами, принятыми в конкретной ОС. Так, для прохождения задачи ОС должна выделить ресурсы: время центрального процессора, оперативную и внешнюю память, необходимые системные программы и устройства ввода — вывода.

Требования на ресурсы оформляются на языке управления заданиями, который позволяет формализовать запросы на ресурсы. Этот язык является внешним языком ОС, обеспечивающим пользователю средство общения с системой.

В качестве ресурсов ВС выступают программные и аппаратные ресурсы. Программные ресурсы включают набор системных программ, составляющих программное обеспечение ЭЦВМ. В зависимости от характера использования программные ресурсы разделяются на однократно используемые, повторно последовательно используемые и многократно параллельно используемые ресурсы.

Однократно используемый ресурс позволяет использовать копию программы только в одной задаче. Если предусматривается повторное использование копии, то ее необходимо загружать.

Повторно последовательно использованный ресурс позволяет использовать программу несколькими задачами последовательно. После работы одной задачи ресурсы этой разновидности восстанавливаются и могут быть доступны другой задаче.

Многократно параллельно используемый ресурс предусматривает одновременное использование программы многими задачами. В этом случае для хранения пере-

менной информации каждой задачи, работающей с этой программой, отводится рабочая область памяти.

Аппаратурными ресурсами являются:

полезное время работы центрального процессора;

оперативная память машины;

внешняя память на магнитных дисках, барабанах и магнитных лентах;

каналы связи с периферийными устройствами;

носители данных на перфокартах и перфолентах практически неограниченной емкости.

Перечисленные ресурсы могут временно сокращаться из-за неисправности отдельных устройств. Способность ОС настраиваться на переменные ресурсы ВС составляет функцию адаптируемости.

Принято различать статическое и динамическое распределение ресурсов, выполняемое ОС.

При статическом распределении ресурсов задача пользователя не принимается системой до тех пор, пока все ее требования на ресурсы не будут удовлетворены. Этот способ распределения ресурсов приводит к их частичному замораживанию, так как задача резервирует выделенные ей ресурсы на весь интервал времени работы. Положительным моментом этого распределения является тот факт, что задача выполняется без задержек.

При динамическом распределении ресурсов требования задач выполняются по ходу их реализации. Может иметь место совместное использование ресурсов, что повышает коэффициент использования ресурса, но может служить причиной задержки прохождения конкретной задачи через ВС.

Одной из существенных функций ОС является обеспечение хранения больших массивов информации при простом доступе.

В современных вычислительных системах предусматривается широкое использование архивов данных, включающих обрабатываемую информацию, программы обработки данных и библиотеки СП, при соответствующем обеспечении автоматизации выполнения основных видов работ по поиску, обмену, контролю и организации хранения.

В этих системах данные рассматриваются отдельно от породивших их задач (программ), т. е. данные суще-

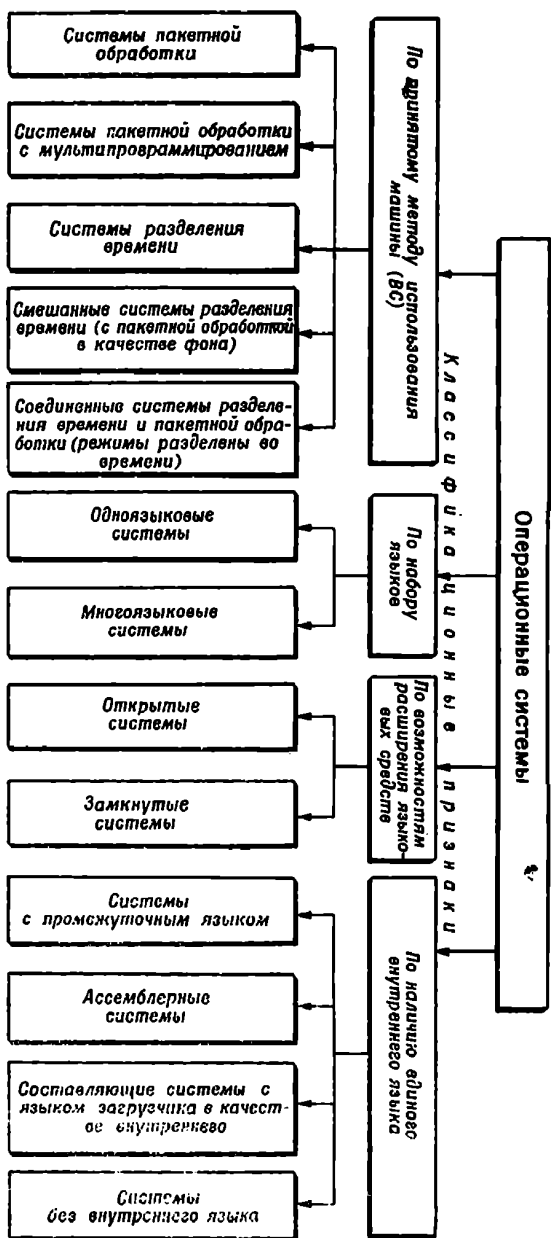


Рис. 4.3. Классификационная схема операционных систем

ствуют независимо, в системе возникает некоторая совокупность таких данных, порождаемых разными задачами. Предусматривается, что данные, полученные в системе, могут быть использованы различными, не обязательно их породившими задачами.

В соответствии с общими принципами построения ОС и исследованиями, приведенными в работе [37], можно представить классификационную схему операционных систем (рис. 4.3).

4.2.2. Основные элементы ОС

Решение общей задачи обеспечения организации работы вычислительной системы можно разделить на следующие части:

- обслуживание ввода — вывода и обменов между различными уровнями памяти;
- планирование работ и распределение ресурсов;
- управление процессом выполнения программ пользователей;
- управление массивами данных (включая и архив);
- управление загрузкой программ;
- связь с оператором и диагностика ошибок;
- ведение протокола работы, учет работ и сбор статистики.

Разнообразие функций, выполняемых ОС, приводит к тому, что в ее состав необходимо включать большое число самых различных программ, работающих в значительной степени асинхронно и независимо друг от друга. Программы, реализующие отдельные функции, называются исполнителями. Ряд сложных функций ОС может быть реализован несколькими исполнителями. В связи с этим в структуре ОС важное место занимает аппарат связи исполнителей и механизм передачи информации от одного исполнителя к другому. Для организации таких связей в составе ОС предусматриваются специальные управляющие программы:

- супервизор задач;
- супервизор ввода—вывода;
- диспетчер заданий;
- главный диспетчер;
- управление данными.

Каждая из перечисленных управляющих программ обеспечивает связь внутри подчиненной группы исполнителей и с другими управляющими программами. Заказ на работу исполнителя всегда оформляется в виде некоторой стандартной заявки и сообщения управляющей программе о заказе. Заявка присоединяется к очереди на обслуживание. При выделении основной памяти для заявки и организации очередей широко используются приемы ассоциативного программирования.

Рассмотрим функции, решаемые отдельными управляющими программами ОС. При этом предположим, что в большинстве случаев перечень функций управляющей программы адекватен списку ее исполнителей.

Супервизор задач выполняет следующие основные функции управления:

- распределение основной памяти;
- загрузка программ в основную память;
- управление параллельным выполнением задач;
- организация службы времени;
- реализация процедур восстановления после возникновения исключительных условий;
- регистрация ошибок;
- сбор итоговых данных об использовании программных средств.

Супервизор ввода — вывода выполняет все функции, прямо или косвенно связанные с операциями ввода — вывода, в частности с передачей данных между основной памятью и устройствами ввода — вывода. Кроме того, он выполняет запоминание и реализацию всех требований по вводу — выводу.

В большинстве операционных систем супервизоры задач и ввода — вывода объединяются в понятие управляющей программы, называемой супервизором. Супервизор получает управление через прерывание. В ряде ОС допускается явное использование требований на обслуживание супервизором с использованием команды «вызов супервизора», которая указывает ссылку на конкретную процедуру обслуживания.

Диспетчер заданий (планировщик) решает задачи обработки некоторой управляющей информации, которая содержит требования к системе и порядок выполнения работ, задаваемых пользователем. Одной из основных целей ОС является максимально возможная загрузка

работой ВС. Это значит, что всякий раз, когда ВС готова выполнить предстоящую работу, то такая работа есть в памяти.

Под заданием понимается полная переработка, включающая одну или несколько связанных программ обработки, таких, как периодический прием информации об объектах, выдача сводки по объектам на заданное время, тестовый контроль за работой устройств. Задание состоит из одного или нескольких шагов задания. Шаг задания — это такой объем работы, с которым связана только одна программа переработки.

Эта информация вводится в систему в форме операторов языка управления заданиями.

Диспетчер заданий реализует следующие функции:

- просмотр введенных данных в целях выделения управляющей информации, интерпретация и анализ управляющих предложений и подготовка необходимых управляющих таблиц, описывающих для системы каждое задание;

- участие в распределении устройств ввода — вывода для заданий;

- выбор задания для выполнения в соответствии с очередью или приоритетностью.

Главный диспетчер (главный планировщик) обеспечивает связь между оператором и системой. Оператор с помощью специальных команд может воздействовать на работу системы. Возможный набор команд управления следующий:

- команды, воздействующие на задание;
- команды, воздействующие на систему;
- команды-запросы на информацию;
- команды-приказы на запись информации.

Команды, воздействующие на задание, вызывают изменение статуса задания, например отмену или временную остановку задания или изменение приоритета задания.

Команды, воздействующие на систему, вызывают изменение действий, осуществляемых диспетчером заданий. Эти команды могут информировать систему о новом устройстве или об устройстве, которое не может более кому-либо распределяться.

Запросы на информацию позволяют оператору запра-

шивать систему о ее статусе или статусе отдельных заданий.

Запросы на информацию позволяют оператору сообщать системе текущую дату и время, вводить информацию в системный журнал и отвечать на системные или программные запросы об информации, появившейся в результате применения макроинструкции записи.

В качестве примера рассмотрим схему работы управляющей программы в режиме пошагового планирования (рис. 4.4) применительно к ОС-360 [44].

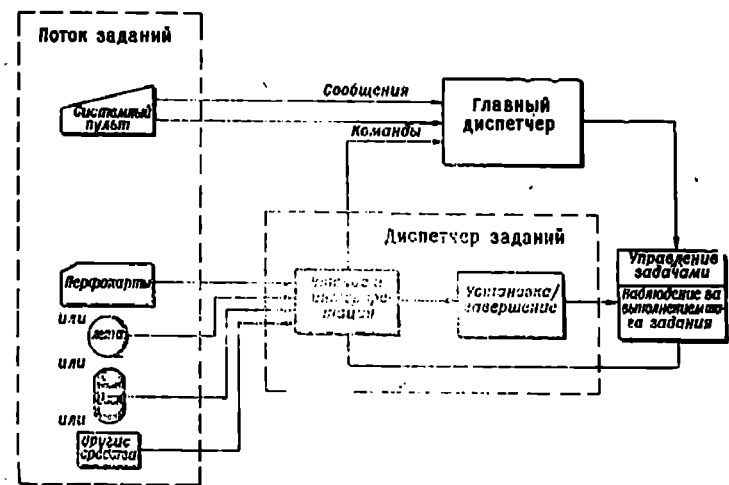


Рис. 4.4. Схема системы последовательного планирования

Поток заданий читается и анализируется диспетчером заданий в пошаговом режиме с перфокарт или перфолент. Программа чтения — интерпретации просматривает предложения для одного шага задания. Программа начальной установки распределяет устройства ввода — вывода и сообщает оператору о носителях информации, которые должны быть установлены, после чего шаг задания будет готов к выполнению. Кроме того, эта программа формирует заказ программам управления задачами проконтролировать выполнение шага задания. Эти программы передают управление модулю загрузки и наблюдают за его работой. Главный диспетчер принимает команды и выполняет действия в соответствии с макрокомандами.

Программы управления данными предназначены для перемещения информации между основной и внешней памятью и для хранения информации во внешней памяти. Эти программы используются для определения местоположения данных, подготовки для них областей основной памяти, для считывания и записи данных.

Для управления потоком данных пользователю разрешается либо составлять свои собственные программы обслуживания каналов, либо пользоваться стандартными программами управления данными, которые предполагают стандартное оформление хранения массивов информации.

Для обеспечения поиска конкретных данных в их совокупности необходимо предусмотреть организационное оформление архива и некоторую схему идентификации хранилищ.

Основу наиболее употребительных схем организации архивов данных составляют фиксированные по объему участки внешней памяти, называемые файлами (хранилищами). Файл представляет собой элемент данных во внешней памяти, в котором кроме собственно данных хранится некоторая управляющая информация (имя файла и его границы во внешней памяти).

Система идентификации файлов, как правило, децентрализована. Для исключения совпадения имен последние разрешается задавать 20—40 символами.

В связи с тем что файл является автономной единицей, существование которой не зависит от наличия какой-либо управляющей информации вне его, возникает необходимость обеспечения установления связи очередной задачи с конкретным файлом (выполнение некоторых начальных процедур поиска и контроля) через операционную систему вычислительной машины. После того как связь с заданным файлом установлена, можно выполнять работу, предусмотренную алгоритмом задачи.

Итак, с точки зрения организации долговременного хранения файлов данных наиболее существенными являются наличие автономной, стандартным образом оформленной единицы данных во внешней памяти и наличие программных модулей, обеспечивающих связь пользователя с необходимыми ему данными.

Возможности, предоставляемые операционной системой пользователю при работе с данными внутри файла,

определяются способом организации файлов, задаваемым при их образовании. Так, например, в операционной системе БЭСМ-6 предусматривается четыре способа организации файлов.

1. Файл-адрес. Пользователь может представлять себе адресный файл, как сплошной массив элементов, пронумерованных подряд, начиная с номера, который пользователь задает при образовании файла. В этом случае обмен между файлом и оперативной памятью разрешается любыми порциями, начиная с любого номера элемента.

2. Файл-запись. Данная группа файлов характеризуется общностью структуры, выбранной в соответствии с рекомендациями международной организации стандартов о структуре файлов на магнитной ленте.

Согласно этим рекомендациям каждый файл представляет собой последовательность записей. Запись есть группа подряд расположенных машинных слов, которая выступает в роли кванта обмена между файлом и оперативной памятью.

Записи объединяются в блоки, которыми и осуществляется обмен, т. е. блок может содержать одну или несколько полных записей и соответствует понятию зоны магнитной ленты или тракту магнитного барабана.

В зависимости от структуры записи различают файл-длину, файл-признак и файл-переменный. Файл-длина представляет собой последовательность записей постоянной длины, задаваемой при его создании. При организации обмена с таким файлом указывается его имя и адрес зоны оперативной памяти приема записи (блока). Переход внутри файла на следующую запись (блок) обеспечивается ОС автоматически. С этой целью ОС хранит в своем поле текущий адрес обмена для каждой записи файла (стрелку файла).

Файл-признак отличается от предыдущего тем, что размер каждой записи определяется признаком конца записи, помещенным в последней ячейке записи.

Файл-переменный обеспечивает смысловую структуру организации записи. Размер записи в этом файле указывается в ячейке непосредственно перед записью.

Как файл-признак, так и файл-переменный предусматривают наличие специальной ссылки на место начала очередной записи в файле.

3. Файлы ввода — вывода. Вычислительная система имеет возможность представить в распоряжение пользователя произвольное количество абстрактных каналов ввода — вывода.

Весь материал, подлежащий вводу за время решения задачи, операционная система предварительно записывает на магнитные ленты. Директивы ввода выполняются так, как если бы пользователь располагал несколькими физическими каналами ввода.

Аналогично выводимый материал во время решения задачи накапливается на магнитных лентах, а по окончании задачи происходит фактическая выдача отдельно от всей информации, которая выводилась по абстрактным каналам в соответствии с их приоритетом.

Абстрактные каналы ввода — вывода принято называть файлами ввода — вывода. Эти файлы должны быть перечислены в паспорте задачи, а файлы вывода могут быть образованы в процессе счета. Все операции с файлами ввода — вывода производятся так же, как и с обычными файлами. Исключением является то обстоятельство, что с файла ввода можно только читать, а в файл вывода можно только записывать данные.

4. Библиотечные файлы. Библиотечные файлы обеспечивают возможность обращения к элементам файлов по имени. Эти файлы состоят из разделов, представляющих собой непрерывную последовательность машинных слов, длина которой не превышает объема оперативной памяти.

Библиотечный файл оформляется в соответствии с общим стандартом, принятым для файлов на внешних носителях.

Пользователь может работать одновременно с любым количеством библиотек, хранящихся в архиве. При этом работа с конкретной библиотекой производится следующим образом:

а) операционная система по заданному имени библиотеки выполняет подготовительные процедуры, необходимые для установления связи с библиотечным файлом; если заданная библиотека имеется в распоряжении системы и не занята в данный момент другим пользователем, операционная система подготовит все необходимое для оперативной работы с ней;

б) по заданному имени раздела пользователь полу-

чает возможность работать с программами этого раздела. Заметим, что внутри одной библиотеки все разделы должны иметь разные имена, а в разных библиотеках имена разделов могут совпадать.

Операционная система оснащена для работы с архивом данных рядом элементов внутреннего математического обеспечения. Можно выделить следующие группы архивных программ:

программы работы с библиотеками: создание библиотеки, открытие библиотеки, закрытие библиотеки, уничтожение библиотеки, реорганизация библиотеки, создание цепочки библиотек;

программы работы с разделами: чтение раздела, запись раздела, удаление раздела, поиск раздела, запись комментария, присвоение разделу альтернативного имени;

вспомогательные программы: начальная организация архива, реорганизация каталога; документирование каталога (по частям или полное документирование), контроль за сохранностью архива данных, копирование архива данных.

Большая часть перечисленных выше элементов ПО архива общеизвестна. Остановимся на элементах, представляющих с нашей точки зрения особый интерес.

Необходимость реорганизации библиотеки возникает тогда, когда исчерпано свободное поле для приема новых разделов. Для этой цели имеется специальная вспомогательная программа, обеспечивающая перезапись содержимого старого библиотечного файла в новый, более длинный файл. При выполнении перезаписи производится соответствующая коррекция каталога библиотеки.

В целях повышения скорости работы с архивами библиотек операционная система по соответствующему заказу организует цепочки библиотек. Одним из примеров организации цепочки библиотек может служить предварительный вызов отдельных наиболее часто встречающихся стандартных программ некоторого класса задач в быструю память.

Программы, записанные на барабане, образуют барабанную библиотеку, которая указывается в цепочке библиотек перед ленточной библиотекой. Несмотря на то что программы, находящиеся на барабане, есть и на лен-

те, они будут вызываться в оперативную память с барабана, так как барабанная библиотека расположена в цепочке первой.

При работе с массивами важно обеспечить надежность их хранения, т. е. защиту массивов от сбоя оборудования, недозволенных действий по записи программами других пользователей. Это обеспечивается рядом способов, среди которых можно отметить периодическое дублирование и создание ключей защиты, идентифицируемых только пользователем этих массивов.

ОС должна следить также за тем, чтобы операции пользователя с массивом соответствовали степени доступности этого массива, устанавливаемые пользователем. Так, если пользователь установил для массива статус «считываемый», то не допускается запись данных в этот массив.

4.2.3. Операционная система ЭЦВМ «Минск-32»

Операционная система ЭЦВМ «Минск-32» представляет собой систему программ, основное назначение которой состоит в эффективном использовании машины и в программной совместимости с машинами типа «Минск-22».

Система ориентирована на многопрограммный режим работы. Допускается работа системы в режиме разделения времени. При этом различаются два уровня приоритетов решаемых задач: срочный и обычный. Для срочной задачи программа должна быть решена в возможно более короткий срок по сравнению с программами, поступающими в обычном порядке. Для обычных программ организуется очередь на их выполнение в порядке поступления задач.

Структура и функции ОС

Основу ОС составляет система программ ДИСПЕТЧЕР, которая предназначена для организации управления прохождением задач на машине. Система программ ДИСПЕТЧЕР представляет собой программное развитие аппаратных возможностей машины и включает следующий набор программ:

- начальный вызов;

- резидент;
- координатор;
- разгрузка лент вывода;
- запись пакета заданий на НМЛ.

Программа **НАЧАЛЬНЫЙ ВЫЗОВ** предназначена для первоначального вызова в оперативную память программ **РЕЗИДЕНТ**, **КООРДИНАТОР** и таблицы внешних устройств машины. При выполнении этой программы в специальные ячейки оперативной памяти заносятся календарная дата и время начала вызова программ. После завершения работы программы включается электронный датчик времени, используемый для учета текущего времени.

Таблица внешних устройств машины хранит информацию о ресурсах ОС (комплектность машины). В минимальный комплект машины, необходимый для нормального функционирования программного обеспечения, входят следующие компоненты:

- вычислитель с МОЗУ объемом 16 384 слова;
- пульт оператора с пишущей машинкой (ПМ);
- лентопротяжный механизм для установки ленты системы, содержащей таблицу внешних устройств, программы **РЕЗИДЕНТ**, **КООРДИНАТОР** и **СОВМЕСТИМОСТЬ**;
- лентопротяжный механизм для установки ленты вывода.

Перечисленный набор не включает устройств, используемых в рабочих программах. Каждая рабочая программа сообщает **ДИСПЕТЧЕРУ** перечень устройств, необходимых для реализации обработки информации. Для программ, написанных для машины «Минск-32» (П-32), в задании на включение указывается только требуемый объем МОЗУ и количество лентопротяжных механизмов НМЛ. Для программ П-22 вместе с перечнем потребных устройств необходимо задавать соответствие между отдельными устройствами машин.

Программа **РЕЗИДЕНТ** постоянно находится в оперативной памяти машины и предназначена для контроля за сохранностью программ **КООРДИНАТОР** и **СОВМЕСТИМОСТЬ** и для повторного вызова этих программ в процессе выполнения рабочих программ.

Программа **КООРДИНАТОР** управляет одновременным выполнением нескольких П-32. При этом может

оказаться, что для одной или нескольких программ нет свободного устройства вывода на перфокарты или устройства широкой печати. В этом случае результаты будут накапливаться на ленте вывода (ЛВ). Вывод результатов с ЛВ на УВК и УПЧ осуществляется специальной программой РАЗГРУЗКА ЛЕНТЫ ВЫВОДА, которая включается в работу по специальной директиве оператора.

КООРДИНАТОР, являясь составной частью **ДИСПЕТЧЕРА**, присутствует в **МОЗУ** в течение всего времени выполнения П-32 и выполняет следующие функции:

- в любой момент работы машины принимает задания на выполнение рабочих программ и ставит их в очередь;

- выполняет учет текущего времени;

- информирует оператора о текущем времени через фиксированные промежутки;

- подсчитывает время, затраченное на выполнение рабочих программ блоками **КООРДИНАТОРА**.

Для выполнения этих функций **КООРДИНАТОР** включает в свой состав следующие программы:

- **МОНИТОР**, обеспечивающий двустороннюю связь оператора с **КООРДИНАТОРОМ**, прием заданий на выполнение рабочих программ, подготовку и запуск рабочих программ на выполнение;

- **СУПЕРВИЗОР**, осуществляющий реакцию **КООРДИНАТОРА** на экстракоды внешних устройств, сбой ВнУ и устройств обмена (УО) и на завершение работы ВнУ;

- **АДМИНИСТРАТОР**, осуществляющий управление ходом выполнения блоков **КООРДИНАТОРА** и рабочей программы и учет времени;

- **АНАЛИЗАТОР СБОЕВ ВЫЧИСЛИТЕЛЯ**, осуществляющий реакцию **КООРДИНАТОРА** на сбой вычислителя.

Взаимосвязь рабочей программы с отдельными частями программы **КООРДИНАТОР** показана на рис. 4.5.

При выполнении рабочих программ возможно их прерывание по концу работы ВнУ. В этом случае управление передается **СУПЕРВИЗОРУ**, который расшифровывает причины прерывания, выполняет необходимые действия и передает управление **АДМИНИСТРАТОРУ** для переключения на рабочую программу.

При прерываниях по сбою устройства обмена и ВнУ управление передается АНАЛИЗАТОРУ СБОЕВ ВЫЧИСЛИТЕЛЯ (в случае сбоев вычислителя) или блоку СБОИ ВнУ и УО из СУПЕРВИЗОРА (в случае сбоев ВнУ и УО). При этом выполняются следующие функции:

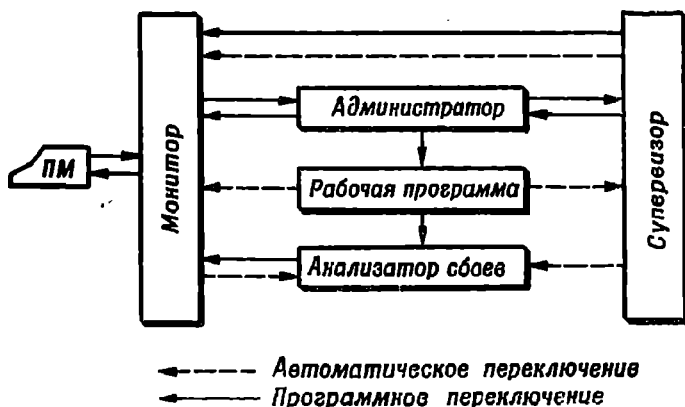


Рис. 4.5. Схема взаимодействия рабочей программы с координатором

- анализируется причина сбоя;
- сообщается оператору о наличии сбоя;
- принимается решение на дальнейшие действия с программой;
- если возможно, то предпринимается попытка повторить выполнение необходимого участка программы;
- осуществляется передача управления АДМИНИСТРАТОРУ для переключения на выполнение рабочей программы.

В процессе работы **КООРДИНАТОР** информирует оператора о различного рода возможных ситуациях с помощью сообщений и указаний, выдаваемых на машинке пульта оператора. Под сообщением понимаются сведения, информирующие о действиях **КООРДИНАТОРА** и рабочей программы. После вывода сообщения соответствующая программа не требует никакого вмешательства оператора, а указания требуют обязательного ответа **КООРДИНАТОРУ**.

На программу, подготавливаемую для машинного ре-

шения, разрабатывается задание, содержащее совокупность сведений, в соответствии с которыми выполняются одна или несколько рабочих программ последовательно.

Задание подготавливается на перфокартах и передается ДИСПЕТЧЕРУ оператором посредством директив, задаваемых с ПМ пульта оператора. КООРДИНАТОР включает задание в очередь на выполнение. Реализация задания производится при условии, что имеется:

- свободный рабочий уровень;
- свободный участок оперативной памяти, равный или больший требуемого для первой рабочей программы задания;
- потребный объем внешней памяти.

Результаты вычислений в ходе работы программы могут быть выведены на внешние устройства, предусмотренные программистом. Если эти устройства заняты, то координатор сообщает об этом оператору. В соответствии с ответом оператора либо запрещается выполнение этой программы, либо продолжается с записью результатов вычислений на ленту вывода.

Совокупность программ, обеспечивающая выполнение рабочих программ в языке ЭЦВМ «Минск-22М» на ЭЦВМ «Минск-32», составляет программную систему совместимости машин. Система совместимости позволяет использовать на ЭЦВМ «Минск-32» все средства программного обеспечения машины «Минск-22».

Программа СОВМЕСТИМОСТЬ выполняет следующие функции:

- ввод и подготовку к выполнению П-22 по управляющей информации, заданной согласно инструкции выполнения этой программы на машине «Минск-22»;
- пуск П-22;
- программную реализацию системы прерывания машины «Минск-22»;
- реакцию на сбой машины «Минск-32» при выполнении П-22;
- реакцию на прерывания по концу работы ВнУ;
- учет времени выполнения П-22;
- реализацию директив оператора по управлению выполнением П-22 и выдачу сообщений и указаний оператору.

Выполнение этих функций производится монитором,

интерпретатором и анализатором сбоев. Взаимосвязь составных частей программы СОВМЕСТИМОСТЬ с П-22 показана на рис. 4.6.

Монитор обеспечивает подготовку к выполнению очередной П-22, переход от выполнения одной программы к другой и связь оператора с СОВМЕСТИМОСТЬЮ путем приема директив от оператора и выдачи ему сообщений и указаний.

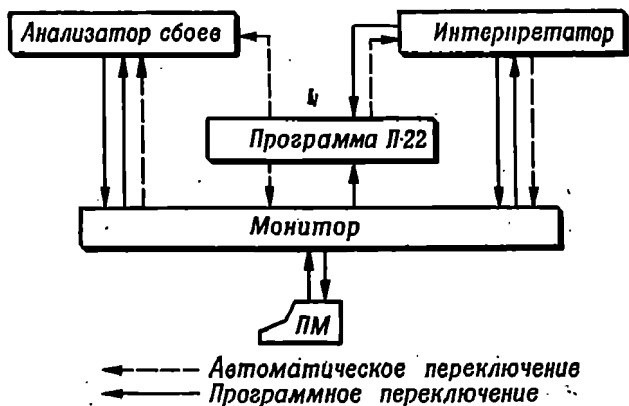


Рис. 4.6. Схема взаимодействия П-22 с программой совместимости

Интерпретатор обеспечивает программную реализацию команд машины «Минск-22», не включенных в систему команд машины «Минск-32», и систему прерывания.

Анализатор сбоя обеспечивает реакцию на сбой вычислителя, устройства обмена и ВнУ, формирование и вывод через монитор на пишущую машинку пульта управления сообщений и указаний о возникших сбоях и определенные действия по устранению последствий сбоя.

Выполнение программ П-22 производится на основании заданий, разрабатываемых согласно инструкции по обработке данных на машине «Минск-22». После приема задания или после выполнения имеющихся в очереди заданий монитор программы совместимости приступает к обработке задания. Обработка задания включает

установку режимов работы внешних устройств, подготовку магнитных лент, ввод П-22 и исходных данных задачи.

После ввода программы и исходных данных оператор одной из специальных директив производит пуск П-22 для выполнения. Все команды П-22 выполняются в соответствующих устройствах машины до выборки команды, не включенной в систему команд машины «Минск-32». Эти команды вызывают прерывание и переход к блоку экстракода из интерпретатора, который производит расшифровку команды и ее программную реализацию.

В процессе интерпретации команд П-22 производится отработка системы прерывания машины «Минск-22» и анализируются сбои вычислителя и внешних устройств. Эти действия выполняются специальными блоками программы совместимости.

Признаком окончания выполнения П-22 является команда останова машины «Минск-22», которая для машины «Минск-32» является экстракодом. Необходимо отметить, что выполнение программ в режиме совместимости производится только в одиночном режиме.

Этапы подготовки и решения задач на машине

Последовательность подготовки и решения задач рассмотрим на основе системы символического кодирования (ССК). Основными этапами подготовки задачи к решению являются трансляция, сборка и загрузка программ (рис. 4.7).

Заказ на трансляцию вводится с перфокарт, перфолент или магнитной ленты по специальной директиве оператора с ПМ. В соответствии с заказом вводится символическая программа (СИМП) и записывается на личную ленту пользователя (ЛМЛ). Для выполнения ввода СИМП в память машины вызывается с ленты системы (ЛС) программа ВКАРТ (ввод карт).

При наличии заказа на трансляцию ОС организует вызов транслятора ССК (ТСК), который и выполняет заданную работу. Транслятор переводит СИМП в программу на языке загрузки и записывает ее на ЛМЛ.

Если в процессе трансляции будет выявлена необхо-

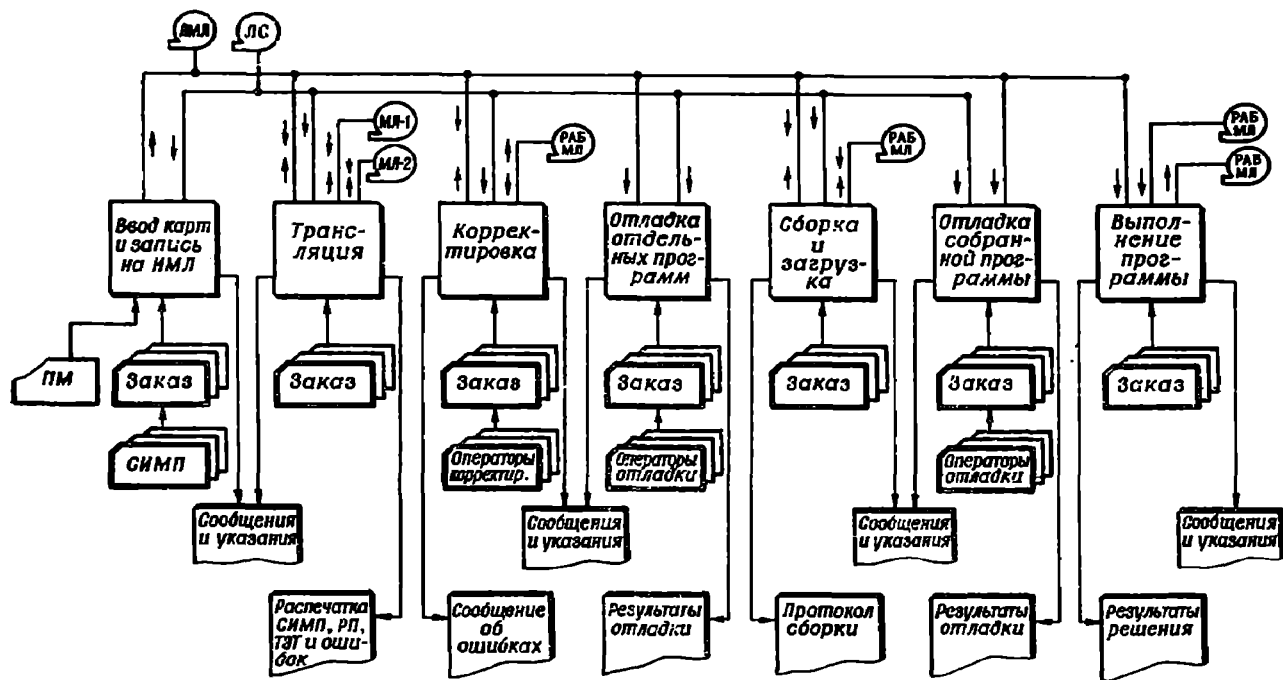


Рис. 4.7. Схема (вариант) последовательности подготовки и прохождения задачи через ЭЦВМ «Минск-32»

димось в корректировке, то при наличии заказа и корректирующей информации эти действия могут быть выполнены специальной корректирующей программой КОРРЕКТОР.

Отладка программ может быть выполнена с помощью специальных отладочных программ ОП1 или ОП2, включенных в программное обеспечение системы.

При сборке к программе на языке загрузки присоединяются ее внутренние программы, хранящиеся как в личной, так и в общей библиотеке в языке загрузки. Результат сборки можно рассматривать как самостоятельную программу на языке загрузки. Собранная программа может быть присоединена к любой составляемой программе наравне с другими библиотечными программами.

Сборка программ допускается и в том случае, если в библиотеке присутствуют не все внутренние программы, потребные в данной СИМП. Недостающие программы присоединяются после их разработки к рабочей программе в процессе повторной сборки.

Загрузку собранных программ осуществляет загрузчик, работающий под управлением координатора. Результатом загрузки является рабочая программа, размещенная в оперативной памяти и готовая к счету.

В процессе трансляции, сборки и отладки СИМП вся необходимая информация выдается на ПМ (сообщения и указания) и на алфавитную печать.

4.3. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СИСТЕМЫ «ОБРАБОТКА НАБЛЮДЕНИЙ»

4.3.1. Математическая модель пакета прикладных программ

При решении некоторых прикладных военных задач приходится сталкиваться с отсутствием полной информации, необходимой для выполнения работы. Так, например, известно изменение с течением времени множества параметров, которые необходимо ввести в модель исследования. Однако их значение может быть неизвестно для того момента времени, в который исследуется операция. Как правило, этот момент времени наступает позже тех, для которых имеется информация.

В этом случае необходимо прогнозирование информации [47].

К такой информации можно отнести например сведения о количестве вооружения в различных частях и подразделениях противника, тактико-технические характеристики образцов вооружения (как своего, так и противника), оперативно-тактические нормативы, характеристику методов применения вооружения и т. д.

Среди этих сведений можно выделить две группы характеристик:

— характеристики, прямо определяемые развитием народного хозяйства и техническим прогрессом и являющиеся в известной мере обобщенными, например, общие затраты на вооружение, показатели оснащенности войск техникой и другие характеристики;

— характеристики, являющиеся производными от первой группы и определяемые не только развитием народного хозяйства и техническим прогрессом, но и потребностями в решении тех или иных задач, а также оптимальным распределением затрат, обоснованным выбором образцов вооружения и методов их использования.

Эти характеристики могут быть получены путем прогнозирования на основе существующих методов математической статистики (методы экспертных оценок, анализа регрессии и корреляции). Любой из этих методов может быть применен для прямого или косвенного прогнозирования. Под прямым прогнозированием понимается совокупность приемов непосредственного определения искомой величины. При косвенном прогнозировании предполагается получение искомой величины на основе прогнозируемых значений вспомогательных величин.

Анализ регрессии и корреляции как статистический метод определения и изменения взаимозависимостей отнюдь не является новой научной дисциплиной. Теоретические проблемы регрессии и корреляции в настоящее время уже настолько разработаны, что внедрение их в статистическую практику и проверка полезности этих методов могут быть самостоятельной серьезной задачей.

Статистический анализ результатов экспериментов (наблюдений) позволяет решить ряд практически важных задач. Во-первых, обеспечивается объективная оценка изучаемого процесса (процессов) по результатам наблюдений. Во-вторых, выявляются статистические зако-

номерности изучаемого процесса с точки зрения получения как количественной, так и качественной оценки. В-третьих, на основе полученных оценок обеспечивается построение моделей прогнозирования и принятия решений.

Наличие большого выбора отечественной и зарубежной литературы по математической обработке результатов наблюдений [26, 32, 41 и др.] позволяет сосредоточить внимание на вопросе разработки программного обеспечения ЭЦВМ, обеспечивающего проведение анализа регрессии и корреляции.

Специалист, работающий в области применения методов анализа регрессии и корреляции с использованием ЭЦВМ, для руководителя «является тем же, чем телескоп для астронома или микроскоп для биолога. До изобретения телескопа астрономы были вынуждены работать в очень сложных условиях и развитие их науки шло очень медленно. Появление телескопа не означает, что нам больше не нужны астрономы, а говорит лишь о том, что у них появилась возможность проникнуть в свой предмет, а следовательно, приступить к изучению значительно более сложных вопросов»¹.

Пакет прикладных программ математической обработки наблюдений должен выполнять следующие функции:

- ввод, редактирование и размещение исходных данных во внешней памяти;

- вывод любого заданного числа строк исходной матрицы данных;

- исключение резко выделяющихся наблюдений на основе анализа одномерных рядов распределения;

- построение одномерных частотных распределений с автоматическим выбором числа интервалов;

- расчет числовых характеристик одномерных рядов распределений: средних арифметических, дисперсий, показателей асимметрии и эксцесса, коэффициентов вариации и т. д.;

- вычисление соответствующих величин для проверки нормальности распределений по различным критериям;

- преобразования над столбцами матрицы исходных

¹ Райветт П., Акофф Р. Л. Исследование операций. М., изд-во «Мир», 1966, с. 134—135.

данных в целях получения по методу наименьших квадратов оценок для коэффициентов регрессии практически любого вида уравнения, линейного относительно оцениваемых параметров;

повторение пунктов 2—6 для преобразованной матрицы;

вычисление коэффициентов регрессии для нормированных и натуральных значений переменных;

получение матриц коэффициентов парной и частной корреляции;

вычисление характеристик, необходимых для проведения статистического анализа уравнения регрессии (множественный коэффициент корреляции, частные и множественные коэффициенты детерминации, остаточная дисперсия, значения t — критерия для коэффициентов регрессии, коэффициенты корреляции между коэффициентами регрессии и другие характеристики);

последовательный отсев несущественных факторов по наименьшему значению t — отношения величины коэффициента регрессии к его среднеквадратичной ошибке или отсев любого из факторов по желанию исследователя, т. е. реализация схемы многошагового регрессионного анализа;

построение доверительных интервалов для расчетных значений функции по уравнению, признанному окончательным в процессе отсева несущественных факторов;

вычисление показателей, характеризующих качество модели: относительная и средневзвешенная ошибки аппроксимации, среднеквадратичное отклонение;

вывод результатов вычислений в заданном объеме в целях документирования.

Х а р а к т е р и с т и к и о д н о м е р н ы х р я д о в р а с п р е д е л е н и я

Основные свойства распределения измеряются соответствующими числовыми характеристиками, обеспечивающими выполнение оценки качества по каждому из одномерных рядов.

Наиболее распространенными числовыми характеристиками рядов распределения являются средняя арифметическая, дисперсия, среднеквадратичное отклонение, коэффициент вариации, показатели асимметрии и эксцесса.

Средняя арифметическая числовая характеристика используется наиболее часто. Выборочное среднее, являющееся приближенной оценкой теоретического среднего (математического ожидания) генеральной совокупности, определяется по формуле

$$x_j = \frac{1}{n} \sum_{i=1}^n x_{ij} \text{ при } j = 1, 2, \dots, k. \quad (1)$$

Средняя арифметическая характеристика как показатель среднего уровня, вокруг которого колеблются значения членов вариационного ряда, является важной, но недостаточной характеристикой распределения. Это уточнение может быть получено при знании среднего квадратичного или стандартного отклонения, определяемого для каждого столбца по формуле

$$S_j = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_{ij} - \bar{x}_j)^2}. \quad (2)$$

Величина

$$S_j^2 = \frac{1}{n} \sum_{i=1}^n (x_{ij} - \bar{x}_j)^2 \quad (3)$$

называется выборочной дисперсией.

Величины S_j и S_j^2 дают хотя и состоятельные, но смещенные оценки для теоретических значений среднеквадратичного отклонения σ и дисперсии σ^2 . В связи с этим при небольших выборках для вычисления применяют несколько измененные выражения, дающие несмещенные оценки для σ и σ^2 , т. е. значения $\bar{S}_j$ и $\bar{S}_j^2$ вычисляются по формулам:

$$\bar{S}_j^2 = \frac{1}{n-1} \sum_{i=1}^n (x_{ij} - \bar{x}_j)^2; \quad (4)$$

$$\bar{S}_j = \sqrt{\bar{S}_j^2}. \quad (5)$$

Оценки среднеарифметической и дисперсии для случайной выборки позволяют при условии нормального распределения утверждать, что в границах, определяе-

мых как $(\bar{x}-\bar{S}, \bar{x}+\bar{S})$ содержится приблизительно 68% всех наблюдений, в границах $(\bar{x}-2\bar{S}, \bar{x}+2\bar{S})$ размещается уже 95% всех наблюдений и почти все наблюдения (99,7%) укладываются в трехсигмовый интервал.

Ошибка среднего значения вычисляется по формуле

$$S_{\bar{x}_j} = \frac{\sigma_j}{\sqrt{n-1}} = \frac{S_j}{\sqrt{n}}. \quad (6)$$

Величина ошибки среднего значения используется для определения радиуса и границ доверительного интервала среднего значения:

$$\Delta \bar{x}_j = t(n-1) S_{\bar{x}_j}; \quad (7)$$

$$x_j^{(1)} = \bar{x}_j - \Delta \bar{x}_j; \quad (8)$$

$$x_j^{(2)} = \bar{x}_j + \Delta \bar{x}_j, \quad (9)$$

где $\Delta \bar{x}_j$ — радиус доверительного интервала среднего значения j -го признака;

$t(n-1)$ — значение t — критерия Стьюдента для $n-1$ степени свободы и выбранного уровня значимости;

$\bar{x}_j^{(1)}, \bar{x}_j^{(2)}$ — левый и правый концы доверительного интервала.

В случае если имеет место значительное отклонение распределения от нормального, эти характеристики не дают качественной оценки. В качестве меры отклонения распределения от нормального используются показатели асимметрии (скошенности) и эксцесса (островершинности).

Эти характеристики получаются расчетным путем на основании следующих зависимостей:

$$K_{1j} = \frac{\sqrt{n(n-1)}}{n-2} \cdot \frac{M_{3j}}{M_{2j}^{3/2}}; \quad (10)$$

$$K_{2j} = \frac{n-1}{(n-2)(n-3)} \left[(n+1) \left(\frac{M_{4j}}{M_{2j}^2} - 3 \right) + 6 \right] \quad (11)$$

на основании работы [41]. Здесь M_{2j}, M_{3j}, M_{4j} — центральные моменты соответственно второго, третьего и четвертого порядка.

Приближенно гипотезу о принадлежности распределения к нормальному можно проверить с помощью показателей асимметрии и эксцесса. Принято считать [26], что если

$$K_1 \leq 3\sigma_{K_1} \text{ и } K_2 \leq 5\sigma_{K_2},$$

то наблюдаемое распределение можно отнести к нормальному.

Согласно работе [41] среднеквадратичные отклонения для характеристик K_1 и K_2 соответственно равны:

$$S_{K_1} = \sqrt{\frac{6n(n-1)}{(n-2)(n+1)(n+3)}}; \quad (12)$$

$$S_{K_2} = \sqrt{\frac{24n(n-1)^2}{(n-3)(n-2)(n+3)(n+5)}}. \quad (13)$$

Из других числовых характеристик часто используется коэффициент вариации, характеризующий степень рассеяния индивидуальных значений членов ряда по отношению к среднему уровню. Коэффициент вариации выражается в процентах и вычисляется по формуле

$$V_j = \frac{\bar{s}_j}{\bar{x}_j} \cdot 100\%. \quad (14)$$

Ошибка коэффициента вариации определяется по формуле

$$\sigma_{V_j} = \frac{V_j}{\sqrt{n}} \sqrt{0,5 + V_j^2}. \quad (15)$$

В связи с тем что нормальное распределение играет особую роль при изучении характеристик одномерных рядов распределения, необходимо включить в алгоритм обработки данных проверку соответствия выборочного распределения нормальной форме. Кроме анализа показателей асимметрии и эксцесса может быть выполнена проверка нормальности распределения на основании критерия Пирсона, Колмогорова и других.

Одним из распространенных критериев является критерий Пирсона (χ^2), вычисляемый по формуле

$$\chi_j^2 = \sum \frac{(m_j - m'_j)^2}{m'_j}, \quad (16)$$

где m_j — эмпирические частоты j -го признака;
 m'_j — теоретические частоты j -го признака.

Критерий χ^2 обычно применяется для оценки дискретных распределений, которые не являются нормальными по своей природе. Однако для практических целей удобно использовать нормальное распределение и для характеристики дискретных распределений, что во многих случаях является оправданным.

Для непрерывных распределений критерий χ^2 может дать неудовлетворительные результаты из-за неудачного выбора числа и длины интервалов, искажающих истинный характер распределения. Поэтому его применение носит ограниченный характер.

В противоположность критерию χ^2 , применение которого связано с группировкой исходных данных по интервалам, критерий Колмогорова основывается на непосредственно наблюдаемых (несгруппированных) значениях величин. Критерий Колмогорова рекомендуется к применению только для проверки непрерывных распределений.

Вычислительная процедура проверки нормальности распределения должна обеспечивать получение характеристик применительно к трем перечисленным выше критериям с тем, чтобы исследователь сам принял окончательное решение о выборе критерия.

На основе полученных характеристик частотных распределений одномерных рядов можно сделать заключение о качестве материала для последующей обработки.

Анализ регрессии и корреляции

Математические методы, с помощью которых исследуются и обобщаются взаимосвязи варьирующих признаков, называются регрессионным и корреляционным анализом. Методы анализа регрессии и корреляции позволяют выявить количественные закономерности и измерить тесноту связей между изучаемыми признаками.

Наиболее простым типом уравнений связи являются линейные уравнения, которые графически изображаются прямой линией. Линейная функция выражает арифметическую пропорциональность: увеличению (уменьшению)

аргумента или фактора на одну единицу всегда соответствует одинаковый прирост (убывание) функции или результатного признака.

На практике встречается большое количество взаимосвязей, которые заключаются в пропорциональном изменении среднего значения зависимого признака с изменением факторного. Если в прямоугольной системе координат нанести точки, ординаты и абсциссы которых будут соответствовать значениям факторного и результатного признаков, то полученная совокупность точек будет представлять поле корреляции.

В корреляционном поле можно провести прямую линию так, чтобы все точки находились по возможности ближе к этой линии, т. е. сумма отклонений была бы минимальной. Уравнение этой линии (линии регрессии) в общем виде можно записать так:

$$y = b_0 + b_1x, \quad (17)$$

где b_0 — длина отрезка, отсекаемого на оси ординат,
 b_1 — тангенс угла наклона линии регрессии к оси абсцисс.

Это уравнение, выражающее общую закономерность взаимосвязей между признаками x и y , называется уравнением регрессии. Для определения коэффициентов уравнения регрессии используется метод наименьших квадратов [19]. В основу этого метода положено требование, чтобы сумма квадратов отклонений фактических величин от расчетных по уравнению была бы минимальной. Математически это требование может быть записано так:

$$Q = \sum_{i=1}^n (y_i - \tilde{y}_i)^2 \rightarrow \min. \quad (18)$$

Для определения констант уравнения прямой $\tilde{y}_i = b_0 + b_1x_i$, отвечающей требованию метода наименьших квадратов, заменим его значением

$$Q = \sum_{i=1}^n (y_i - b_0 - b_1x_i)^2.$$

Полученное выражение будет принимать минимальное значение при

$$\text{т. е.} \quad \frac{dQ}{db_0} = 0 \quad \text{и} \quad \frac{dQ}{db_1} = 0,$$

$$\left. \begin{aligned} \frac{dQ}{db_0} &= -2 \sum_{i=1}^n (y_i - b_0 - b_1 x_i); \\ \frac{dQ}{db_1} &= -2 \sum_{i=1}^n (y_i - b_0 - b_1 x_i) x_i \end{aligned} \right\}$$

или

$$\begin{aligned} \sum_{i=1}^n y_i &= nb_0 + b_1 \sum_{i=1}^n x_i; \\ \sum_{i=1}^n y_i x_i &= b_0 \sum_{i=1}^n x_i + b_1 \sum_{i=1}^n x_i^2. \end{aligned} \quad (19)$$

Система уравнений (19), в которой b_0 и b_1 являются неизвестными, называется системой нормальных уравнений для выбранного уравнения регрессии (17). После решения системы уравнений (19) относительно b_0 и b_1 уравнение регрессии (17) становится вполне определенным.

В качестве меры колеблемости фактических величин вокруг расчетных, т. е. ошибки оценки, принимается остаточное среднеквадратичное или стандартное отклонение фактических величин y_i от расчетных $\tilde{y}_i$. Стандартную ошибку оценки по уравнению регрессии, вычисляемую по данным генеральной совокупности, обозначают через σ_{yx} :

$$\sigma_{yx} = \sqrt{\frac{\sum_{i=1}^n (y_i - \tilde{y}_i)^2}{n}}. \quad (20)$$

Вычисление среднеквадратичной ошибки выборки производится по формуле

$$S_{yx} = \sqrt{\frac{\sum_{i=1}^n (y_i - \tilde{y}_i)^2}{n-2}}. \quad (21)$$

где $(n-2)$ — число степеней свободы.

Потеря двух степеней свободы вызвана использованием двух условий при определении уравнения регрессии. Для больших выборок $n \approx n-2$ и потерю степеней свободы можно не учитывать. Тогда

$$\tilde{S}_{yx} = \sqrt{\frac{\sum_{i=1}^n (y_i - \tilde{y}_i)^2}{n}}. \quad (22)$$

Остаточное среднеквадратичное отклонение используется для построения доверительных границ для оценок по уравнению регрессии. Для этих целей выбирается вероятность p_1 в качестве гарантии того, что оцениваемая величина y не будет ниже оценки $\tilde{y}$ более, чем на $t_{p_1} S_{yx}$, и выбирается вероятность p_2 , которая гарантирует верхнюю границу для $\tilde{y}$. Тогда верхняя доверительная граница запишется $(\tilde{y} + t_{p_1} S_{yx})$, а нижняя — $(\tilde{y} - t_{p_2} S_{yx})$. Величины t_{p_1} и t_{p_2} находятся по выбранным значениям p_1 и p_2 по таблицам t — критерия Стьюдента, если считать распределение вероятностей ошибок $y - \tilde{y}$ близким к распределению Стьюдента [46].

Тогда доверительные границы можно записать в виде следующего неравенства:

$$b_0 + b_1 x_i - t_{p_1} S_{yx} \leq y_i \leq b_0 + b_1 x_i + t_{p_2} S_{yx}.$$

В случае если $p_1 = p_2$, доверительные границы выражаются симметричными линиями относительно линии регрессии. Установленные доверительные границы отражают неточность предсказания результатного признака, обусловленную необъясненной вариацией, так как связь не функциональная, а корреляционная.

В работе [26] доказано, что

$$\sigma_y^2 = \sigma_{\tilde{y}}^2 + \sigma_{yx}^2, \quad (23)$$

т. е. общая дисперсия состоит из двух частей. При этом $\sigma_{\tilde{y}}^2$ объясняется регрессией и определяется

$$\sigma_{\tilde{y}}^2 = \frac{\sum_{i=1}^n (\tilde{y}_i - \bar{y})^2}{n}. \quad (24)$$

Разложение суммы квадратов отклонений на составные части дает возможность разработать методы оцен-

ки, известные под названием дисперсионного анализа [26]. Для выборочных данных равенство (23) становится приближенным:

$$S_y^2 \approx \sigma_y^2 + S_{yx}^2 \quad (25)$$

Остаточное среднеквадратичное отклонение выражается в тех же единицах, что и первоначальные значения y_i . В связи с этим для различных уравнений стандартные ошибки оценки по уравнениям несопоставимы. Для сопоставления используются отвлеченные показатели взаимосвязей.

Одним из интересных показателей, характеризующих долю объясненной дисперсии во всей дисперсии, служит отношение

$$R^2 = \frac{\sigma_y^2}{S_y^2}, \quad (26)$$

называемое коэффициентом детерминации.

В качестве относительной меры того, насколько изменение зависимой переменной обусловлено влиянием изменения аргумента, рассматривается коэффициент корреляции

$$R = \sqrt{\frac{\sigma_y^2}{S_y^2}}.$$

Из уравнения (25) следует, что

$$\sigma_y^2 = S_y^2 - S_{yx}^2.$$

Следовательно, коэффициент корреляции

$$R = \sqrt{1 - \frac{S_{yx}^2}{S_y^2}}. \quad (27)$$

В общем случае коэффициент корреляции может принимать значения в диапазоне $-1 \leq R \leq 1$.

Высокое по модулю значение коэффициента корреляции соответствует более тесной взаимосвязи зависимой переменной и аргумента.

Стандартная ошибка коэффициента регрессии в вы-

борках, выполненных в нормально распределенной или близкой к ней совокупности, оценивается по формуле

$$S_b = \frac{S_{yx}}{S_x \sqrt{n}}. \quad (28)$$

Доверительные границы для уровней вероятности p_1 и p_2 можно записать так:

$$b_1 - S_b t_{p_1} \leq \beta \leq b_1 + S_b t_{p_2},$$

где β — коэффициент регрессии генеральной совокупности.

Доверительные границы индивидуального предсказания зависимой переменной можно определить для уровней вероятности p_1 и p_2 так:

$$\tilde{y}_{x_i} - t_{p_1} S_{\tilde{y}_{x_i}} \leq \tilde{y}_{x_i} \leq \tilde{y}_{x_i} + t_{p_2} S_{\tilde{y}_{x_i}}.$$

Стандартная ошибка коэффициента корреляции при больших выборках оценивается по формуле

$$\sigma_r = \frac{1 - \rho^2}{\sqrt{n - 1}},$$

где ρ — коэффициент корреляции генеральной совокупности. В связи с тем что значение ρ не известно, его заменяют коэффициентом корреляции, вычисленным по выборочной совокупности

$$S_R = \frac{1 - R^2}{\sqrt{n - 1}}. \quad (29)$$

При определении существенности коэффициента корреляции выдвигается гипотеза, по которой коэффициент корреляции генеральной совокупности ρ принимается равным нулю и определяется, совместимо ли это предположение с выборочным коэффициентом корреляции. Если $\rho = 0$, то

$$\sigma_r = \frac{1}{\sqrt{n - 1}}.$$

Для проверки нулевой гипотезы рекомендуется [21] зависимость

$$\mu = \frac{|R|}{\sigma_r}. \quad (30)$$

Если $\mu \geq 2,58$, то гипотеза отвергается, корреляция признается существенной. Если $\mu < 2,58$, то нулевая гипотеза принимается и взаимосвязь признается ненадежной.

Метод наименьших квадратов является основным методом анализа регрессии и корреляции. Одним из вспомогательных методов анализа регрессии и корреляции является метод моментов. В этом методе применяются другие зависимости, которые дают результаты, соответствующие методу наименьших квадратов, но более удобные при составлении программы на ЭЦВМ. При использовании метода моментов расчеты ведутся без учета потерь степеней свободы, поэтому оценки S_y , S_x , S_{yx} получаются смещенными.

Если выборка большая, то $n \approx n-1$, если же маленькая, то S_y , S_x , S_{yx} уточняются с помощью специальных поправочных коэффициентов. При условии, что начало отсчета взято в точке с координатами $\bar{x}$ и $\bar{y}$, нормальные уравнения (19) примут вид:

$$\left. \begin{aligned} \sum_{i=1}^n (y_i - \bar{y}) &= nb_0 + b_1 \sum_{i=1}^n (x_i - \bar{x}); \\ \sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x}) &= \\ &= b_0 \sum_{i=1}^n (x_i - \bar{x}) + b_1 \sum_{i=1}^n (x_i - \bar{x})^2, \end{aligned} \right\} \quad (31)$$

но так как $\sum_{i=1}^n (x_i - \bar{x}) = 0$ и $\sum_{i=1}^n (y_i - \bar{y}) = 0$, то из первого уравнения системы получим, что $b_0 = 0$.

Второе уравнение имеет вид

$$\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x}) = b_1 \sum_{i=1}^n (x_i - \bar{x})^2.$$

Коэффициент регрессии в этом случае определяется так:

$$b_1 = \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^n (x_i - \bar{x})^2}. \quad (32)$$

Вычисление коэффициента корреляции с учетом несмещенных оценок $\hat{S}_{yx}^2$ и σ_y^2 можно выполнить по формулам:

$$\hat{S}_{yx}^2 = \frac{\sum_{i=1}^n y_i^2 - b_0 \sum_{i=1}^n y_i - b_1 \sum_{i=1}^n x_i y_i}{n}; \quad (33)$$

$$\sigma_y^2 = \frac{\sum_{i=1}^n y_i^2}{n} - \bar{y}^2. \quad (34)$$

Значение коэффициента детерминации имеет вид

$$R^2 = \frac{b_0 \sum_{i=1}^n y_i + b_1 \sum_{i=1}^n x_i y_i - n \bar{y}^2}{\sum_{i=1}^n y_i^2 - n \bar{y}^2}. \quad (35)$$

Если начало координат расположено в точке $(\bar{x}, \bar{y})$, то $\bar{y}^2=0$ и формула (35) примет вид

$$R^2 = \frac{b_1 \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (y_i - \bar{y})^2}. \quad (36)$$

Значение b_1 можно подставить из формулы (32). Тогда

$$R = \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}. \quad (37)$$

Из определения дисперсии следует, что

$$\sum_{i=1}^n (y_i - \bar{y})^2 = n \sigma_y^2;$$

$$\sum_{i=1}^n (x_i - \bar{x})^2 = n \sigma_x^2.$$

Следовательно,

$$R = \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})}{n\sigma_x\sigma_y}. \quad (38)$$

Средняя величина произведений $(y_i - \bar{y})(x_i - \bar{x})$, определенная для $i = 1, 2, \dots, n$, называется ковариацией или первым смешанным моментом:

$$C_{yx} = \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})}{n}. \quad (39)$$

Отсюда

$$R = \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})}{n\sigma_x\sigma_y}. \quad (40)$$

Эта зависимость является центральной формулой при вычислении показателей корреляции по способу моментов. Формулу ковариации (39) можно преобразовать к виду, удобному для вычислений:

$$C_{yx} = \frac{\sum_{i=1}^n y_i x_i}{n} - \bar{y} \bar{x}. \quad (41)$$

Из зависимости (36) можно получить значение b_1 :

$$b_1 = \frac{R^2 \sum_{i=1}^n (y_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})}.$$

Используя зависимость (40), можно получить

$$b_1 = R \frac{\sigma_y}{\sigma_x}. \quad (42)$$

Полученная зависимость также является одной из основных формул метода моментов. Зависимость для оп-

ределения b_0 получается из первого уравнения системы нормальных уравнений (19):

$$b_0 = \frac{\sum_{i=1}^n y_i}{n} - b_1 \frac{\sum_{i=1}^n x_i}{n}. \quad (43)$$

Для вычисления стандартной ошибки оценки по уравнению регрессии преобразуется зависимость (27) в вид

$$S_{yx} = \sigma_y \sqrt{1 - R^2}. \quad (44)$$

Стандартные ошибки и доверительные границы коэффициента регрессии, корреляции, среднего и индивидуального предсказания находятся по тем же формулам, что и при методе наименьших квадратов.

Если выборка небольшая, а необходимо иметь точные оценки доверительных границ, стандартные отклонения и ошибки уточняются с учетом потерь степеней свободы. При этом используются следующие зависимости:

$$\begin{aligned} S_y^2 &= \sigma_y^2 \frac{n}{n-1}; \\ S_x^2 &= \sigma_x^2 \frac{n}{n-1}; \\ \bar{S}_{yx}^2 &= S_{yx}^2 \frac{n}{n-2}; \\ R &= \sqrt{1 - \frac{S_{yx}^2}{\sigma_y^2} \cdot \frac{n-1}{n-2}}. \end{aligned}$$

При статистическом анализе регрессии и корреляции приходится выяснять взаимосвязи не двух, а трех и более признаков. Применение метода линейной регрессии и корреляции и изучение взаимосвязей всех признаков попарно в этом случае может оказаться недостаточным, так как при исследовании взаимосвязи между зависимой переменной и одним фактором не позволяет выявить ее в чистом виде.

Взаимосвязь трех и более признаков часто выража-

ется с помощью линейного уравнения регрессии. Линейное уравнение множественной регрессии отличается от уравнения простой регрессии дополнительными коэффициентами, число которых определяется количеством независимых переменных. В общем случае при K независимых переменных уравнение регрессии имеет вид

$$\tilde{y} = b_0 + b_1 x_1 + b_2 x_2 + \dots + b_k x_k. \quad (45)$$

Система нормальных уравнений для множественной регрессии будет иметь вид:

$$\left. \begin{aligned} \sum_{i=1}^n y_i &= b_0 n + b_1 \sum_{l=1}^n x_{i1} + b_2 \sum_{i=1}^n x_{i2} + \\ &\quad + \dots + b_k \sum_{i=1}^n x_{ik}; \\ \sum_{i=1}^n y_i x_{i1} &= b_0 \sum_{i=1}^n x_{i1} + b_1 \sum_{i=1}^n x_{i1}^2 + b_2 \sum_{i=1}^n x_{i1} x_{i2} + \\ &\quad + \dots + b_k \sum_{i=1}^n x_{i1} x_{ik}; \\ \\ \sum_{i=1}^n y_i x_{ik} &= b_0 \sum_{i=1}^n x_{ik} + b_1 \sum_{l=1}^n x_{il} x_{ik} + \\ &\quad + b_2 \sum_{l=1}^n x_{i2} x_{ik} + \dots + b_k \sum_{l=1}^n x_{lk}^2. \end{aligned} \right\} \quad (46)$$

В результате решения системы нормальных уравнений относительно $b_0, b_1, b_2, \dots, b_k$ получаются значения коэффициентов уравнения множественной регрессии. Следовательно, для нахождения коэффициентов уравнения необходимо иметь суммы данных всех признаков, суммы их квадратов и суммы всех парных произведений.

Систему нормальных уравнений для вычисления коэффициентов множественной регрессии можно составить и на основе расчетных величин — сумм отклонений от среднеарифметических, дисперсий и ковариаций, коэффициентов парной корреляции. Так, при вычислениях по

отклонениям от средних арифметических система нормальных уравнений имеет вид:

[illegible]

После соответствующих упрощений система (47) может быть решена методом обратных матриц [19]. Уравнение множественной линейной регрессии не выражает функциональной зависимости, при которой каждой комбинации факторов соответствовала бы строго определенная величина результатного признака. Значение этого признака, определяемого по уравнению множественной регрессии, при любой комбинации факторов $x_1, x_2, \dots, x_k$ является лишь наиболее вероятной величиной ожидаемого результата. Следовательно, необходимо вы-

числить ошибку оценки по уравнению и меру тесноты связи.

Стандартную ошибку оценки по уравнению множественной регрессии можно вычислить следующим образом:

$$S_{y.12\dots k} = \sqrt{\frac{\sum_{i=1}^n (y_i - \tilde{y}_i)^2}{n - k - 1}}. \quad (48)$$

В качестве относительной меры того, насколько изменение зависимой переменной обусловлено влиянием изменения аргументов, принято рассматривать коэффициент множественной корреляции

$$R_{y.12\dots k} = \sqrt{1 - \frac{S_{y.12\dots k}^2}{S_y^2}}. \quad (49)$$

Доверительные границы оценки множественной регрессии и индивидуального предсказания оцениваются аналогично случаю простой линейной регрессии, только t_p берется для $n - k - 1$ степеней свободы.

При изучении влияния нескольких факторов на результатный признак большое значение имеет частная регрессия и корреляция. В этом случае выясняется взаимосвязь, которая существует между одним из факторов и результатом при условии, что остальные факторы остаются неизменными. Для получения уравнения частной регрессии в уравнение множественной регрессии вместо неизменных факторов подставляют их средние значения, тогда

$$\tilde{y}_{j.12\dots k} = b_0 + b_1 \bar{x}_1 + b_2 \bar{x}_2 + \dots + b_k \bar{x}_k. \quad (50)$$

Коэффициенты частной регрессии совпадают с соответствующими коэффициентами множественной регрессии.

Коэффициенты частной корреляции показывают тесноту связи между результатным признаком и каким-либо факторным признаком при закреплении на неизменном уровне значений других факторов.

Расчетная зависимость для получения коэффициентов частной корреляции имеет вид

$$R_{\nu_{j.12\dots(j-1)(j+1)\dots k}} = \frac{S_{y.12\dots(j-1)(j+1)\dots k}^2 - S_{y.12\dots k}^2}{S_{y.12\dots(j-1)(j+1)\dots k}^2}. \quad (51)$$

Знаменатель дроби представляет собой остаточную дисперсию результирующего признака после того, как уже было учтено влияние всех факторов, кроме j -го. Второй член числителя — это остаточная дисперсия при учете влияния признака $\bar{x}_j$. Числитель показывает снижение остаточной дисперсии за счет добавления к факторам и фактора x_j .

4.3.2. Алгоритмы основных программ пакета

Пакет прикладных программ анализа регрессии и корреляции включает управляющую программу, набор модулей обработки информации и обслуживающие программы.

Управляющая программа пакета обеспечивает ввод управляющих карт задания с контролем правильности и формирование цепочки модулей, обеспечивающих обработку информации в соответствии с заказом. Блок-схема управляющей программы показана на рис. 4.8.

Рассмотрим алгоритмы основных программ пакета.

Программы модулей пакета разработаны исходя из предположения, что матрица исходных данных (наблюдений) расположена в памяти машины по столбцам:

$$\begin{array}{ccccccc} y_1 & x_{11} & x_{12} & \dots & x_{1p}; \\ y_2 & x_{21} & x_{22} & \dots & x_{2p}; \\ y_3 & x_{31} & x_{32} & \dots & x_{3p}; \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ y_n & x_{n1} & x_{n2} & \dots & x_{np}. \end{array}$$

Предварительная обработка наблюдений предусматривает выполнение следующих действий:

- подготовка исходных данных к работе с модулями пакета:

— исключение резко выделяющихся наблюдений с последующим преобразованием матрицы исходных данных.

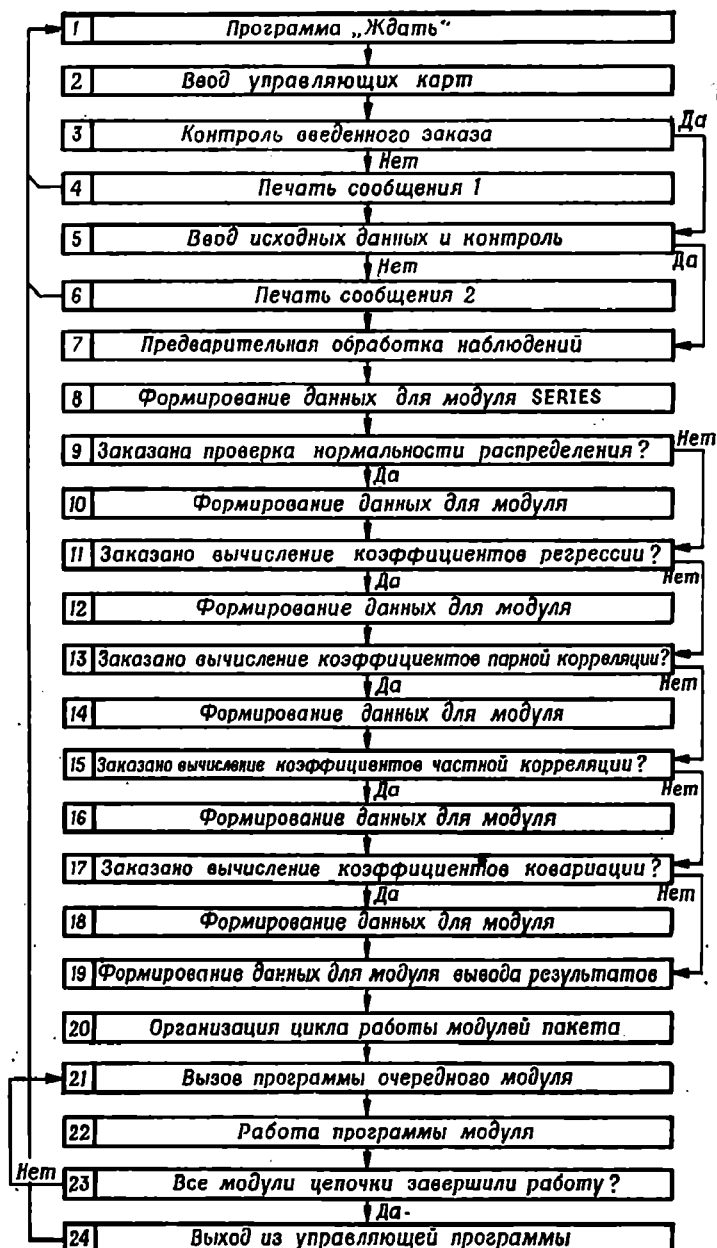


Рис. 4.8. Укрупненная блок-схема пакета программ

Матрица исходных данных может быть подготовлена к вводу по строкам. В этом случае перед работой программ пакета необходимо выполнить процедуру REBUILD1, перестраивающую исходную матрицу к виду «по столбцам»:

```

procedure REBUILD1 (A, B, N, P);
value N, P; integer N, P; array A, B;
  begin integer I, J, K; K := 1;
    for J := 1 step 1 until P do
      for I := 1 step 1 until N do
        begin B[K] := A[I, J]; K := K + 1 end
      end;

```

Возможно задание исходных данных по столбцам с контрольной суммой по каждому из столбцов в целях последующей проверки правильности подготовки информации. Действия по проверке контрольных сумм и перестройке исходного массива выполняются процедурой REBUILD2:

```

procedure REBUILD2 (A, B, N, P);
value N, P; integer N, P; array A, B;
  begin integer I, J, K, R, H; real C;
    R := 0; for J := 1 step 1 until P do
      begin C := 0; for I := 1 step 1 until N do
        C := C + A[I, J];
        if C ≠ A[I + 1, J] then
          begin for I := 1 step 1 until N do
            print (A[I, J]); K := I + 1;
            print (A[K, J], C); R := 1 end
          end; if R = 1 then stop (0,1); H := 0; K := 1;
          for J := 1 step 1 until P do
            begin for I := 1 step 1 until N do
              begin B[K] := A[I + H, J]; K := K + 1 end;
              H := H + 1
            end
          end;

```


В случае несовпадения суммы, отперфорированной после столбца, и суммы, полученной программой, производится печать этого столбца и двух сумм. При отсутствии ошибок в подготовке данных выполняется перестройка матрицы исходных данных в соответствии с принятым форматом.

В случае если массив исходных данных построен по строкам с контрольной суммой на каждую строку, то необходимо выполнить процедуру REBUILD3:

```

procedure REBUILD3 (A, B, N, P);
  value N, P; integer N, P; array A, B;
  begin integer I, J, K, R; real C; R := 0;
    for I := 1 step 1 until N do
  begin C := 0; for J := 1 step 1 until P do
    C := C + A[I, J];
    if C ≠ A[I, J + 1] then
      begin for J := 1 step 1 until P do
        print (A[I, J]); K := I + 1;
        print (A[I, K], C); R := 1
      end
    end; if R = 1 then stop (0,1); K := 1;
    for J := 1 step 1 until P do
      for I := 1 step 1 until N do
        begin B[K] := A[I, J]; K := K + 1
      end
    end;
  end;

```

После выполнения перестройки в целях получения заданного информационного формата матрица данных готова для обработки.

Перед тем как приступить к обработке статистических данных, полученных экспериментально, необходимо убедиться в том, что матрица *X* не содержит грубых ошибок, которые могли бы привести к существенному искажению результатов.

Выполняя просмотр столбцов матрицы X , можно обнаружить наблюдения, резко выделяющиеся по своему значению среди других элементов вектора-столбца. Однако следует заметить, что выявление ошибок, за исключением явных промахов, представляет значительные трудности. Иногда такие наблюдения могут быть предусмотрены физической сущностью явления и их исключение может оказаться нежелательным с профессиональной точки зрения исследователя. В ряде случаев вообще нельзя утверждать заранее, следует ли считать резко выделяющееся наблюдение ошибкой без учета взаимосвязей этой переменной с другими переменными (факторами) модели. Окончательное заключение по качеству каждого из факторов модели можно дать на основании анализа всех результатов обработки.

При предварительной обработке можно использовать для принятия решения об исключении резко выделяющихся наблюдений один из известных статистических критериев, основанных на распределении крайних членов вариационного ряда в случайной выборке, извлеченной из нормальной генеральной совокупности [26, 46, 49 и др.]. Несмотря на то что нормальное распределение допускает появление сколь угодно больших отклонений от среднего значения, вероятность появления их в выборке ограниченного объема очень мала. Поэтому, задаваясь достаточно малой вероятностью q совершить ошибку, исключая наблюдения, которые на самом деле являются верными, можно построить критические границы для крайних членов вариационного ряда в зависимости от объема выборки n . Вероятность q принято называть уровнем значимости, а вероятность $p=1-q$ доверительной вероятностью, коэффициентом доверия или коэффициентом надежности.

Применение того или иного критерия в каждом конкретном случае зависит от полноты априорных сведений о генеральной совокупности, на основании которой получена выборка. Наиболее характерной является ситуация, когда исследователь не располагает дополнительными сведениями о генеральной совокупности кроме тех, которыми обладает матрица исходных данных X . В этом случае для оценки резко выделяющихся наблюдений каждого столбца исходной матрицы можно воспользоваться табличными значениями распределения:

$$\xi(\bar{x}_j, S) = \max_i \frac{|x_{ij} - \bar{x}_j|}{S_j},$$

где

$$\bar{x}_j = \frac{1}{n} \sum_{i=1}^n x_{ij};$$

$$S_j^2 = \frac{1}{n} \sum_{i=1}^n (x_{ij} - \bar{x}_j)^2$$

при $j=1, 2, \dots, k$ есть выборочное среднее и выборочная дисперсия.

Представительная таблица $\xi(\bar{x}, S)$ содержится в работе [4].

Алгоритм автоматического исключения резко выделяющихся наблюдений из матрицы исходных данных состоит в следующем. Для каждого из столбцов матрицы ошибки выявляются отдельно. Прежде всего вычисляются значения $\bar{x}_j$ и S_j . Затем выполняется поиск того наблюдения x_{ij} , для которого величина

$$\frac{|x_{ij} - \bar{x}_j|}{S_j}$$

является наибольшей. Если полученная величина больше значения $Z_{q,n}$, заданного на основании таблицы распределения величины $\xi(\bar{x}, S)$, то запоминается соответствующее значение i наблюдения x_{ij} .

Для оставшихся $K-1$ наблюдений исследуемого столбца снова вычисляются $\bar{x}_j$ и S_j . Процедура запоминания элементов вектора-столбца повторяется до тех пор, пока не будут найдены все наблюдения, выходящие за установленные границы:

```

procedure DAIT ( $A, N, P, Z, K$ ); value  $N, P$ ;
    integer  $N, P, K$ ;
    real  $Z$ ; array  $A$ ;
begin array  $A1[1:N]$ ; Boolean array  $B$ ; integer  $i, j$ ;
    real  $R, x1, x2$ ;
    for  $I := 1$  step 1 until  $N$  do  $B[I] := \text{false}$ ;
    for  $j := 1$  step 1 until  $P$  do
    begin  $x1 := x2 := 0$ ; for  $i := 1$  step 1 until  $N$ 

```

```

do  $x1 := x1 + A[i, j]$ ;
 $x1 := x1/N$ ; for  $i := 1$  step 1 until  $N$ 
do  $x2 := x2 + (A[i, j] - x1)^2$ ;
 $x2 := x2/N$ ; for  $i := 1$  step 1 until  $N$  do  $A1[i] := 0$ ;
for  $i := 1$  step 1 until  $N$  do
begin if  $B[i]$  then goto  $M1$ ;  $A1[i] := ABC(A[i, j] - x1)/x2$ ;
 $M1 : \text{end}$ ;  $M2 : R := -999\,999\,999$ ;
for  $i := 1$  step 1 until  $N$  do if  $A1[i] > R$  then begin
 $R := A1[i]$ ;  $K := i$  end;
if  $R > Z$  then begin  $A1[K] := 0$ ;  $B[K] := \text{true}$ ; goto  $M2$ 
end end;
comment перекомпоновка матрицы;  $K := 0$ ;
for  $i := 1$  step 1 until  $N$  do
begin if  $B[i]$  then goto  $M3$ ;  $K := K + 1$ ;
for  $j := 1$  step 1 until  $P$  do  $A[K, j] := A[i, j]$ ;  $M3 : \text{end}$ 
end;

```

После завершения работы по всем столбцам из матрицы исходных данных исключаются те строки, в которых были обнаружены резко выделяющиеся наблюдения.

В связи с тем что наименьшей из возможных дисперсий обладает только нормальное распределение, для несимметричных наблюдений установленные критические границы могут значительно перекрывать диапазон допустимых значений либо справа, либо слева. В этом случае будут исключены наблюдения, находящиеся с одной стороны вариационного ряда, а наблюдения с противоположной стороны ряда останутся, несмотря на то, что достоверность некоторых из них также может вызывать сомнение. Для исключения этого явления следует сначала преобразовать исходное распределение в нормальное (если это возможно), а затем выполнять описанный выше алгоритм.

Анализ регрессии и корреляции начинается с получения характеристик одномерных рядов распределения, расчетные зависимости для которых приведены форму-

лами (1—15). Действия по вычислению характеристик реализуются процедурой SERIES:

```

procedure SERIES (A, P, N, y, y1, y2, y3, y4, x, x1, x2, S,
                  S1, S2, V, V1);
value P, N; integer P, N; array A, y, y1, y2, y3, y4, x, x1,
                  x2, S, S1, S2, V, V1;
    begin integer I, J; real T;
      for J := 1 step 1 until P + 1 do
        begin y [J] := 0; for I := 1 step 1 until N do
          y [J] := y [J] + A [I, J]; y1 [J] := y [J] / N
        end;
      for J := 1 step 1 until P + 1 do
        begin x1 [J] := 0; for I := 1 step 1 until N do
          x1 [J] := x1 [J] + (A [I, J] - y1 [J]) ↑ 2;
          x2 [J] := x1 [J] / N; x [J] := SQRT (x2 [J]);
          S2 [J] := x1 [J] / (N - 1); S [J] := SQRT (S2 [J]);
          S1 [J] := S [J] / SQRT (N); INTERPOLATION (N, T);
          y2 [J] := T × S1 [J]; y3 [J] := y1 [J] - y2 [J];
          y4 [J] := y1 [J] - y2 [J]; V [J] := S [J] / y1 [J];
          V1 [J] := V [J] / SQRT (N) × SQRT (5 + (V [J]) ↑ 2)
        end
      end;

```

Близость фактического распределения к нормальному можно проверить на основании критерия Пирсона по соответствующей процедуре:

```

procedure PIRSON (A, N, x, K);
value N; integer N, K; array A; real x;
begin integer I, J; real D, E, P, H, L, M, PI, S;
    array T [1 : D]; integer array B, Q [1 : D];
    PI := 3.1416; D := LN (N) / LN (2); E := 999 999 999;
    P := -999 999 999; S := 0;
    for I := 1 step 1 until N do
      begin S := S + A [I]; if A [I] < E then E := A [I];
        if A [I] > P then P := A [I] end;
      H := (P - E) / (D + 1); L := S / N; S := 0;

```

```

    for  $I: = 1$  step 1 until  $N$  do
 $S: = S + (A[I] - L) \uparrow 2$ ;  $P: = \text{SQRT}(S/(N - 1))$ ;
    for  $J: = 1$  step 1 until  $D$  do
 $B[J]: = 0$ ; for  $I: = 1$  step 1 until  $N$  do
        begin  $M: = 0$ ;
            for  $J: = 1$  step 1 until  $D$  do
begin if  $E + M \leq A[J] \wedge A[J] < E + M + H$  then goto  $M1$ ;
                 $M: = M + H$  end;
                 $M1: B[J]: = B[J] + 1$ 
                end;  $M: = 0$ ;
            for  $J: = 1$  step 1 until  $D$  do
begin  $T[J]: = E + H/2 + M$ ;  $M: = M + H$  end;
                for  $J: = 1$  step 1 until  $D$  do
 $Q[J]: = H \times N/P/(\text{SQRT}(2 \times P)) \times \text{EXP}(- (T[J] -$ 
                     $- L) \uparrow 2/2/P)$ ;  $x: = 0$ ;
                    for  $J: = 1$  step 1 until  $D$  do
 $x: = x + (B[J] - Q[J]) \uparrow 2/Q[J]$ ;  $K: = N - 2$ 
                        end;

```

После расчета характеристик одномерных рядов и проверки нормальности распределения можно получить уравнение регрессии, которое содержало бы одну зависимую переменную y и P — аргументов (факториальных признаков). Вместо абсолютных значений вычисления ведутся в отклонениях от арифметических средних каждой переменной, что позволяет уменьшить размер матрицы системы за счет исключения параметра b_0 .

Составление и решение системы нормальных уравнений выполняется с помощью процедуры *MATRICA*, а обращение матрицы — с помощью процедуры *INVERT* [49].

После решения системы вычисляются по формулам (47—50) расчетные значения результатного признака, сумма квадратов отклонений, эмпирическая остаточная дисперсия, среднее квадратическое отклонение без учета потерь степеней свободы, квадрат ошибки оценки по уравнению и ошибка оценки по уравнению множественной регрессии, несмещенная оценка дисперсии среднего зна-

чения результатного признака и ряд других показателей.

Следующим этапом анализа регрессии и корреляции является вычисление коэффициентов парной корреляции, их ошибок, критерия значимости и доверительных интервалов для коэффициентов. Эти действия выполняются процедурой *TWINCOR*:

```

procedure TWINCOR (A, N, P, y1, z0, z2, R3, SR, TR1,
                    TR2, z12, z21, RKM1, RKM2);
value N, P; integer N, P; array A, y1, z0, z2, R3, SR, TR1,
                    TR2, z12, z21, RKM1, RKM2;
    begin integer I, J, K;
    for J := 1 step 1 until P + 1 do
        for I := 1 step 1 until N do
            z0 [I, J] := A [I, J] - y1 [J];
        for J := 1 step 1 until P + 1 do
            for K := 1 step 1 until P + 1 do
                begin z2 [J, K] := 0; for I := 1 step 1 until N do
                    z2 [J, K] := z2 [J, K] + z0 [I, J] × z0 [I, K]
                end;
                for J := 1 step 1 until P + 1 do
                    for K := 1 step 1 until P + 1 do
                        begin R3 [J, K] := z2 [J, K] / SQRT (z2 [I, J] × z2 [K, K]);
                            SR [J, K] := (1 - R3 [J, K] / SQRT (N - 1));
                            TR1 [J, K] := R3 [J, K] / SR [J, K];
                            TR2 [J, K] := R3 [J, K] / SQRT (1 - R3 [J, K] ↑ 2) ×
                                × SQRT (N - 2);
                            z12 := .5 × LN ((1 + R3 [J, K] / (1 - R3 [J, K])) -
                                - R3 [J, K] / 2 / (N - 1) - 1.96 / SQRT (N - 3));
                            z21 := .5 × LN ((1 + R3 [J, K] / (1 - R3 [J, K])) -
                                - R3 [J, K] / 2 / (N - 1) + 1.96 / SQRT (N - 3));
                            RKM1 [J, K] := (EXP (2 × z12) - 1) / (EXP (2 ×
                                × z12) + 1);
                            RKM2 [J, K] := (EXP (2 × z21) - 1) / (EXP (2 ×
                                × z21) + 1)
                        end
                    end;
    end;

```

Далее необходимо рассчитать коэффициенты частной корреляции и частной детерминации, а также их оценки. Эти действия выполняются процедурой *PARTCOR*:

```

procedure PARTCOR  (A, P, N, C1, Q, RYK, RYK2, SR,
                     TR3, TR4, y1, E, E0);
value P, N; integer P, N; real E; array A, C1, Q, RYK,
                     RYK2, SR, TR3, TR4, y1, E0;
begin integer I, J, K, L; real zy; zy := 0;
      for I := 1 step 1 until N do
        zy := zy + (A[I, 1] - y1[1]); L := 2;
        for J := 2 step 1 until P + 1 do
begin MATRICA (A, C1, Q, P, L); E0[J] := zy;
      for K := 2 step 1 until P + 1 do
        begin if K = L then goto M1;
        E0[J] := E0[J] - C1[K] × Z[1, K];
        M1: end; L := L + 1
      end;
      for J := 2 step 1 until P + 1 do
begin RYK2[J] := 1 - E/E0[J]; RYK[J] := SQRT
      (RYK2[J]);
      SR[J] := (1 - RYK2[J]/SQRT (N - P)); TR3[J] :=
      RYK[J]/SR[J]; TR4[J] := RYK[J]/SQRT
      (1 - RYK2[J]) × SQRT (N - P - 1)
    end
  end;

```

Коэффициенты ковариации, а также параметры доверительных интервалов для расчетных значений результатного признака, т. е. стандартные ошибки расчетных значений и концы доверительных интервалов, рассчитываются с помощью процедуры *COVARIATION*. При этом используется процедура *INTERPOLATION* для определения табличных значений параметра t , соответствующих данному числу степеней свободы:


```

procedure COVARIATION (A, N, P, Q, COV, y1, y2, SY,
                        S4, S6, y5, y6, y7);
value N, P; integer N, P; array A, Q, COV, y1, y2, SY,
                        y5, y6, y7;
                        real S4, S6;
begin integer I, J, K; real T1; array z0[1 : N, 1 : P + 1];
      INTERPOLATION (N, T1);
      for I := 2 step 1 until P + 1 do
        for J := 2 step 1 until P + 1 do
          COV[I, J] := S4 × Q[I, J];
          if P > 8 then goto M1;
          for J := 1 step 1 until P + 1 do
            for I := 1 step 1 until N do
              z0[I, J] := A[I, J] — y1[J];
              for I := 1 step 1 until N do
                begin SY[I] := S6; for J := 2 step 1 until P + 1 do
                  for K := 2 step 1 until P + 1 do
                    SY[I] := SY[I] + COV[J, K] × z0[I, J] × z0[I, K]
                    end;
                  for I := 1 step 1 until N do
                    begin y5[I] := T1 × SY[I]; y6[I] := y2[I] — y5[I];
                      y7[I] := y2[I] + y5[I]
                      end;
                    M1 : end;

```

В состав пакета прикладных программ анализа регрессии и корреляции включены вспомогательные программы обслуживания вывода результатов расчета на различные внешние устройства, программы дублирования пакета программ на различные носители информации и программа контроля правильности хранения информации во внешней памяти.

4.3.3. К вопросу о практическом применении пакета прикладных программ

В последние годы вышло в свет несколько монографий и пособий, содержащих детальное изложение методов анализа регрессии и корреляции. Опубликованы

интересные работы, посвященные применению этих методов в различных задачах.

Во многих вычислительных центрах и научно-исследовательских институтах разработаны эффективные алгоритмы и библиотечные программы анализа регрессии и корреляции на ЭЦВМ. Но, несмотря на имеющиеся теоретические разработки и определенное программное обеспечение, методы анализа регрессии и корреляции еще редко применяются для практических целей.

Внедрение этих методов в практику зависит от следующих условий:

имеются ли проблемы, которые нерационально решать или невозможно решить традиционными методами анализа, но можно успешно решить методами регрессии и корреляции;

имеется ли достоверная исходная информация для количественного и качественного изучения интересующих взаимосвязей;

имеется ли необходимая подготовка у специалистов, которые должны практически использовать методы анализа регрессии и корреляции, а также предпосылки для преодоления некоторого психологического барьера, который обычно возникает при внедрении любого нового метода;

имеется ли достаточная надежность получаемых показателей, чтобы использовать их как верную базу для оценки фактического положения и принятия решений.

Условия, названные в первых трех пунктах, на наш взгляд не препятствуют немедленному внедрению методов анализа регрессии и корреляции в практику.

Вопросы устойчивости показателей регрессий и корреляций в настоящее время изучены слабо. В первом приближении такие показатели могут быть вычислены на основе статистических характеристик наблюдений (коэффициентов регрессии, средних и индивидуальных предсказаний) и их стандартных ошибок. Значительное влияние на величину доверительных интервалов оказывает постановка эксперимента по сбору наблюдений и точность фиксации результатов, а также отбор факторов для многошагового, многофакторного регрессионного анализа.

Необходимость и целесообразность применения методов анализа регрессии и корреляции, а также разработ-

ки специальных пакетов прикладных программ достаточно убедительно показана в ряде публикаций. Рассмотрим в укрупненном виде пример анализа зависимости расхода запасных частей к тракторам от их выработки и расхода топлива [39]. При изложении этого примера попытаемся показать некоторые возможности по обоснованию принятия решений на базе статистической обработки наблюдений.

Для успешного использования тракторного парка страны важное значение имеет рациональная эксплуатация тракторов, поддержание их в работоспособном состоянии путем проведения своевременного ремонта и постоянного технического обслуживания.

При решении вопроса об общем потребном числе запасных частей и о их правильном территориальном распределении необходимо выявить зависимость расхода запасных частей от факторов работы. В качестве решающих факторов в проведенном исследовании были приняты годовая выработка и удельный годовой расход топлива, потребляемого трактором.

На выработку тракторов, удельный расход топлива, износ деталей и затраты на ремонт оказывают влияние многочисленные факторы в разных вариантах. При планировании указанных величин для различных условий эксплуатации не представляется возможным учитывать влияние каждого из многочисленных факторов. Статистические методы анализа регрессии и корреляции обеспечивают выявление совместного результативного действия на основе наблюдений.

С точки зрения подхода к решению задачи представляет интерес группа факторов, оказывающих влияние на интенсивность и характер износа и расхода деталей. К этим факторам отнесены следующие:

конструктивные, связанные с кинематической схемой трактора, с величиной зазоров между сопрягаемыми деталями, числом оборотов или поступательных движений в единицу времени, системой смазки, температурным режимом работы и т. д.;

технологические: чистота поверхности сопряженных деталей, точность обработки поверхностей, колебания твердости, неравномерность покрытий, погрешности обработки и т. д.;

металлургические: изменчивость химического состава, внутренние трещины, шлаковые включения и т. д.;
эксплуатационные: изменения в режиме работы, количество и качество регулировок, рельеф местности, длина гона, число и продолжительность запусков, число и продолжительность простоев, изменение нагрузки, своевременность и качество технического обслуживания, качество горючего и смазочных материалов и т. д.;

почвенные и климатические условия: механический состав почвы, характеристики влажности и обрабатываемости почвы, количество осадков, окружающая температура и т. д.

Приведенный перечень факторов, влияющих на интенсивность и характер износа и на срок службы деталей, показывает их обилие и случайный характер воздействия применительно к множеству машин, работающих на полях страны. Исследование рассматриваемых признаков производилось по данным, взятым относительно разных районов страны.

При построении изучаемых моделей приняты следующие обозначения:

x — годовая выработка трактора мягкой пахоты, га;
 y — удельный годовой расход топлива мягкой пахоты, кг/га;

z — весь годовой расход топлива на всю годовую выработку трактора, кг;

v — удельный годовой расход деталей (запасных частей), оцениваемый в рублях на 1 га мягкой пахоты;

u — удельный расход на ремонт в рублях на 1 га мягкой пахоты.

Связи между перечисленными признаками изучались на основе анализа регрессии и корреляции: найдены уравнения регрессии, исчислены коэффициенты корреляции и корреляционные отношения для числовой характеристики тесноты связи между признаками. Полученные результаты применительно к модели ДТ-54 приведены в табл. 7.

Таким образом, статистическое исследование связей между выработкой, удельным расходом топлива, всей величиной расхода топлива, удельным расходом на ремонт и удельным расходом на запасные части позволяет сделать следующие выводы.

Результаты исследования связи между признаками

№ по пор.	Признаки		Уравнение связи	Характеристики тесноты связи				Единицы совокупности
	функция	фактор (аргумент)		r	σ_r	γ	c_η	
1	z	x	$z_x = 3,747 + 0,00757x$	0,91	0,014	0,93	0,011	Области, края и республики СССР ($n = 151$)
2	y	x	$y_x = 8,85 + \frac{1825}{x}$			0,76	0,029	То же ($n = 152$)
3	y	x	$y_x = 7,5 + \frac{3150}{x}$			0,74	0,013	Украинская ССР ($n = 1265$)
4	u	x	$u_x = 0,45 + \frac{4500}{x}$			0,71	0,061	Области, края и автономные республики РСФСР ($n = 67$)
5	v	x	$v_x = 0,8 + \frac{1375}{x}$			0,73	0,030	Области, края и республики СССР ($n = 143$)
6	v	y	$v_y = -1,936 + 0,406y$	0,63	0,040			То же ($n = 136$)

Между расходом топлива и выработкой трактора существует прямая линейная зависимость, определяемая уравнением

$$z_x = b_0 + b_1 x.$$

Между годовой выработкой, с одной стороны, и удельным расходом топлива, удельным расходом на ремонт, удельным расходом на запасные части — с другой, существует в среднем обратная связь, характеризующаяся гиперболической зависимостью.

Между удельным расходом запасных частей и удельным расходом топлива существует в среднем прямая линейная зависимость.

Удельный расход запасных частей на планируемый период следует определять одновременно по планируемой выработке и удельному расходу топлива для заданного периода.

Автор приведенного исследования указывает на ряд практических приложений полученных результатов.

По данным плановой выработки и планового удельного расхода топлива можно вычислить значение удельного расхода запасных частей для ремонта тракторов различных марок на планируемый период. На основе этого показателя может быть вычислена общая сумма затрат на запасные части, потребные на планируемый период. Полученные данные обеспечивают прогнозирование потребности в запасных частях по различным районам страны. Кроме того, полученные зависимости позволяют получить коэффициенты сменности для большинства деталей трактора с учетом районов страны.

Результаты анализа дают также возможность дифференцировать межремонтные сроки работы машин, так как эти сроки в основном определяются сроками службы деталей.

Полученные формулы связей позволяют наметить также пути косвенной оценки износа деталей по выработке и удельному расходу топлива для исследуемых моделей машин. При наличии целенаправленных исследований и результатов наблюдений по износу деталей возможно принятие обоснованных решений на внесение изменений в конструктивную схему машины.

Из приведенного примера видны широкие возможности применения анализа регрессии и корреляции. Эти

методы уже сейчас могут быть внедрены при определении предсказаний (прогнозов) результативных признаков для оцениваемых элементов наблюдаемых совокупностей. Информационное содержание получаемых показателей будет в значительной степени способствовать научному обоснованию области применения тех или других методов анализа и прогнозирования.

ЛИТЕРАТУРА

1. Агеев М. И., Алик В. П., Галис Р. М. Алгоритмы (1—50). М., ВЦ АН СССР, 1966.
2. Алферова З. В., Волович М. Сортировка информации на ЭВМ. М., «Статистика», 1965.
3. Бокарев В. А. Философские вопросы кибернетики и ее значение в военном деле. М., Воениздат, 1970.
4. Большой Л. Н., Смирнов Н. В. Таблицы математической статистики. М., «Наука», 1965.
5. Венецкий И. Г., Кильдышев Г. С. Основы математической статистики. М., «Статистика», 1970.
6. Гладун В. П., Рабинович З. Л. Быстродействующие алгоритмы сортировки в оперативной памяти. Киев, «Кибернетика», 1, 1965.
7. Глушков В. М., Гладун В. П., Лозинский Л. С., Погребинский С. Б. Обработка информационных массивов в АСУ. Киев, «Наукова думка», 1970.
8. Ершов А. П. Доклад на второй Всесоюзной конференции по программированию. Труды ВКП-2. Вып. 1. Новосибирск, 1970.
9. Ершов А. П. [и др.]. АЛГОЛ-60. Пересмотренное сообщение. М., «Мир», 1965.
10. Зубов В. С. К вопросу о классификации методов внутренней сортировки. Сб. «Вычислительная и организационная техника в проектировании и строительстве». Вып. 4—2. ГИПРОТИЗ, И., 1968.
11. Иванов Д. А., Савельев В. П., Шеманский П. В. Основы управления войсками. М., Воениздат, 1971.
12. Калужнин Л. А., Стогний А. А., Стойкова Л. С. О принципах построения машинных словарей. В сб. «Прикладная лингвистика и машинный перевод». Киев, 1962.
13. Китов А. И. Программирование информационно-логических задач. М., «Советское радио», 1967.
14. Королев М. А., Кузьмин К. С., Лавров С. С., Летичевский А. А., Столяров Г. К., Шура-Бура М. Р. Сообщение об алгоритмическом языке АЛГЭК. Киев, «Кибернетика», 2, 1966.
15. Королев М. А. Обработка экономической информации на электронных машинах. М., «Экономика», 1966.
16. Криницкий Н. А. Таблицы объектно-характеристические. Энциклопедия «Автоматизация производства и промышленная электроника», т. 3, 1964.

17. Курбаков К. И. Кодирование и поиск информации в автоматическом словаре. М., «Советское радио», 1968.
18. Лавров С. С., Гончарова Л. И. Лекции по автоматической обработке данных. М., ВЦ АН СССР, 1969.
19. Линник Ю. В. Метод наименьших квадратов и основы математико-статистической теории обработки наблюдений. Изд. 2-е. М., Физматгиздат, 1962.
20. Лозинский Л. С. Анализ методов внешней сортировки, использующей слияние. Киев, «Кибернетика», 1, 1968.
21. Лукомский Я. И. Теория корреляции и ее применение к анализу производства. Изд. 2-е. М., Госстатиздат, 1961.
22. Мудров В. И. Алгоритм нумерации сочетаний. Вычислительная математика и математическая физика, 1965, т. 5, 4, с. 776—778.
23. Никитин А. И., Шолмов Л. И. Об одном алгоритме сортировки на магнитных лентах. Киев, «Кибернетика», 6, 1966.
24. Попов Г. Х. Техника управления. М., Изд-во МГУ, 1968.
25. Пржиялковский В. В., Смирнов Г. Д., Пыхтин В. Я. Электронная вычислительная машина «Минск-32». М., «Статистика», 1972.
26. Пустыльник Е. И. Статистические методы анализа и обработки наблюдений. М., «Наука», 1968.
27. Разумов О. С. [и др.]. Программирование военных задач для ЭЦВМ. Книга 1. М., Воениздат, 1969.
28. Романов А. Н., Фролов Г. А. Под ред. Дружинина В. В. Основы автоматизации систем управления. М., Воениздат, 1971.
29. Общеотраслевые руководящие методические материалы для создания АСУП. Сборник. М., Гос. комитет Сов. Мин. СССР по науке и технике, 1971.
30. Техничко-эксплуатационные характеристики электронных, клавишных и перфорационных вычислительных машин. Сборник. М., ЦСУ СССР, Главное управление вычислительных работ, 1971.
31. Система математического обеспечения к машине «Минск-32». Библиотека программ. Т. 1—3. М., НИИ ЦСУ СССР, 1970.
32. Смирнов Н. В., Дунин-Барковский И. В. Курс теории вероятностей и математической статистики для технических приложений. Изд. 3-е. М., «Наука», 1969.
33. Солнышков Ю. С. Количественное обоснование решений. М., «Знание», 1971.
34. Спарынкин В. С., Стронгин Р. Г. Оценка эффективности одного метода сортировки на ЦВМ с использованием внешней памяти. Киев, «Кибернетика», 6, 1967.
35. Стецюра Г. Г. Новый принцип построения ЗУ. ДАН СССР. Т. 132, 6, 1960, с. 1291—1294.
36. Ткаченко П. Н. [и др.]. Математические модели боевых действий. М., «Советское радио», 1969.
37. Труды 1-й Всесоюзной конференции по программированию. Киев, 1969.
38. Труды 2-й Всесоюзной конференции по программированию. Новосибирск, 1970.
39. Вопросы баланса народного хозяйства и корреляции. Ученые записки по статистике. Т. 4. М., Изд-во Академии наук СССР, 1959.
40. Фальк В. Н. Алгоритм адресной сортировки. В сб. «Вы-

числительная техника и программирование». Вып. 1. М., «Советское радио», 1966.

41. Фишер Р. А. Статистические методы для исследователей. М., Госстатиздат, 1958.

42. Фишер Ф. П., Сундл Д. Ф. Системы программирования. М., «Статистика», 1971.

43. Фогараш Б. Логика. М., Изд-во иностр. лит., 1959.

44. Функциональная структура ОС/360. М., «Советское радио», 1971.

45. Хотяшов Э. Н., Дудкин Г. Е. Система организации постоянной информации предприятия при использовании ЭВМ. М., «Статистика», 1971.

46. Худсон Д. Статистика для физиков. М., «Мир», 1967.

47. Чув Ю. В. Исследование операций в военном деле. М., Воениздат, 1970.

48. Шураков В. В. Подпрограммы обработки данных. М., «Статистика», 1967.

49. Яноши Л. Теория и практика обработки результатов наблюдений. М., «Мир», 1968.

ОГЛАВЛЕНИЕ

	<i>Стр.</i>
Введение	3
Глава 1. Основные характеристики военной информации, принципов и средств ее обработки	11
1.1. Общие понятия	—
1.2. Классификация	21
1.3. Средства автоматизации	24
1.3.1. Общие сведения	—
1.3.2. Электронная цифровая вычислительная машина (ЭЦВМ) «Минск-32»	29
1.3.3. Единая система электронных вычислительных машин (ЕС ЭВМ)	48
1.4. Организация обработки военной информации в типовом военном вычислительном центре (ТВВЦ)	61
1.4.1. Назначение, функции и структура ТВВЦ	—
1.4.2. Функции основных подразделений ТВВЦ по технологическому процессу обработки информации	70
1.4.3. Методика определения количества ЭЦВМ для ТВВЦ	74
Глава 2. Общие принципы подготовки военных задач к решению на ЭЦВМ	78
2.1. Методика разработки задач	—
2.1.1. Понятие о постановке задачи	—
2.1.2. Математическое описание (модель) задачи	80
2.1.3. Понятие об алгоритме решения задачи	85
2.1.4. Понятие о программе решения задачи на ЭЦВМ	87
2.2. Основные принципы алгоритмизации задач	90
2.2.1. Алгоритм и его основные свойства	—
2.2.2. Алгоритмический процесс	93
2.2.3. Основные законы (принципы) логики	95
2.2.4. Последовательность построения алгоритмов	99
2.3. Краткая характеристика основного алгоритмического языка (АЛГОЛ-60)	102
2.3.1. Общие сведения об алгоритмических языках	—
2.3.2. Назначение и структура АЛГОЛ-60	104
2.3.3. Понятие об АЛГОЛ-программе и реализации ее на ЭЦВМ	108
2.3.4. Элементы программирования на основном алгоритмическом языке	115

	Стр.
Глава 3. Основные методы и приемы обработки больших массивов военной информации на ЭЦВМ	151
3.1. Классификация основных методов обработки больших массивов информации на ЭЦВМ	152
3.1.1. Вычислительные и логические процессы обработки информации	—
3.1.2. Методика выбора и оценки методов логической обработки	156
3.2. Организация хранения больших массивов данных	160
3.2.1. Основные принципы хранения информации	—
3.2.2. Табличный способ хранения информации	163
3.2.3. Хранение информации в виде списков	174
3.2.4. Хранение массивов в библиотеке	184
3.2.5. Организация массивов информации в ЭЦВМ «Минск-32»	185
3.3. Процедуры работы с массивами	192
3.3.1. Сортировка информации	—
3.3.2. Выборка информации	209
3.3.3. Поиск информации	213
3.3.4. Редактирование информации	224
3.4. Сегментация больших массивов, хранящихся во внешней памяти	233
3.4.1. Общие сведения	—
3.4.2. Сегментация для устройств со свободным доступом	236
3.4.3. Сегментация для устройств с зонным доступом	238
Глава 4. Основные принципы программного обеспечения систем обработки военной информации на ЭЦВМ	241
4.1. Состав и характеристика программного обеспечения современной ЭЦВМ	243
4.1.1. Общее программное обеспечение	245
4.1.2. Специальное программное обеспечение	257
4.2. Операционные системы	260
4.2.1. Общие принципы построения ОС	262
4.2.2. Основные элементы ОС	268
4.2.3. Операционная система ЭЦВМ «Минск-32»	276
4.3. Программное обеспечение системы «Обработка наблюдений»	284
4.3.1. Математическая модель пакета прикладных программ	—
4.3.2. Алгоритмы основных программ пакета	304
4.3.3. К вопросу о практическом применении пакета прикладных программ	315
Литература	322

Олег Сергеевич Разумов, Виктор Владимирович Шураков

**Основные принципы и методы обработки военной
информации на ЭЦВМ**

Редактор В. Т. Емельянов

Технический редактор Е. К. Коновалова

Корректор Г. И. Селиванова

Г-70620. Сдано в набор 28.10.74 г. Подписано к печати 18.03.75 г.

Формат 84×103¹/₂ мм. 10¹/₄ печ. л., 17,22 усл. печ. л., 16,605 уч.-изд. л.

Типографская бумага № 2 Тираж 9000

Изд. № 5/5760

Цена 94 коп.

Зак. 1008

Воениздат

103160, Москва, К-160

1-я типография Воениздата

103006, Москва, К-6, проезд Скворцова-Степанова, дом 3